

**Disclosure of the Invention**

The invention encompasses the combination of two printing technologies into a single web fed printing press 10, particularly the combination of flexographic 14 and intaglio 17 printing. Other features may be present in the combination web printing press 10 as well, such as die cutting, offset printing, gravure printing, hologram application, and the like. The aforementioned combining process is accomplished through the use of precision servo motors 35 coupled with motion control hardware 40 and software (Appendix). The computer programs (Appendix) incorporated into this application were written by the inventor to accomplish the complex process of marrying two dissimilar printing processes in a single press.

The combination of two dissimilar printing processes is a complex problem of motion control. Web tension and position must be maintained throughout the printing process in order to achieve accurate print registration. The inventor has implemented precision control of the web using high accuracy servo motors 35 combined with low inertia drive rollers 25, 39, 42. A multi-processor motion control system 40 commands the activity of the servo motors 35 and monitors the resultant movement with digital position feedback. All solenoids, lights, and switches are interfaced to the press computer 40 using an integrated programmable logic controller.

The press controller 40 is comprised of a PC based computer using RISC based motion control boards. Each RISC based motion control board has its own processor, and supports up to eight channels of motion control. A single servo motor 35 defines a channel. Each point of web control uses a unique servo motor 35. Each motion control board runs a unique software program created by the inventor. The operator interface is driven by the host PC running Microsoft Windows and an application program (Appendix). This application program obtains information from the operator, converts it to machine commands, and passes it to the motion control boards via the PC back plane. It is the host PC application that coordinates the entire operation of the press 10, including programmable logic controller functions.

**Brief Description of the Drawing**

FIG. 1A is a diagrammatic side elevational view of flexographic and intaglio print units according to the present invention; and

FIG. 2A is a diagrammatic side elevational view of optional finishing or processing units disposed downstream of the components shown in FIG. 1A.

**Best Mode For Carrying Out The Invention**

As illustrated in FIGS. 1A and 1B, the present combination flexographic and intaglio printing press, generally designated 10, can be looked at as a multi-axis robot. Robots are thought to handle materials in an orderly way, moving them from one place to another, and performing operations along the way to alter the effect of the material. The press handles substrate or web material 11 wound in a roll 12, typically on a three inch cardboard core with an outside diameter of forty inches. The number of feet of material 11 on the roll depends on the type and thickness of the material being printed. The press is not limited to printing on paper; films, polymers, and the like are also suitable as a printing substrate 11. The width of the substrate or web 11 is a function of the design width of the combination press. An unwinder 13 acts as a delivery device for the substrate 11. The substrate 11 is pulled from the unwinder 13 by a first flexographic print unit 14 in the press line. The unwinder 13 includes a conventional tensioning mechanism 16 to provide resistance. The degree of resistance applied to the web by the unwinder 13 is adjustable, and regulated by a stand-alone tension controller.

All of the printers in the press line are flexographic print units 14. Preferably, the flexographic press units 14 precede the intaglio print units 17. One or more flexographic print units 14 apply ink to the substrate 11. The units' motion can be effected as a group with a single servo 35, or independently with a servo 35 for each print unit 14. The servo(s) 35 are controlled by a motion controller board 45 in the press computer 40. Since axis or unit numbers are generally assigned chronologically from the first unit in the line up, the first flexo 14, or group of flexo units 14 are assigned unit 1. This or these servos receive speed and position information from the motion controller 45. A reference signal to the motion controller(s) is derived from one of two sources: 1) in Flexo only mode, where the press is printing with the Intaglio unit disabled, the reference signal is synthesized from internal reference generator software, and 2) in combination mode, the reference signal is generated by an optical encoder 21 mounted on the intaglio unit drive 22. The motion controller 45 generates a command to the servo based on the reference signal generated and other parametric data such as operator data, print repeat length, and so on. Print registration (the alignment of one printed feature to another) can be adjusted by actions of the operator making entries to the host computer, which is translated to

machine commands, and ultimately affects the command to the servo(s). The printed ink is dried at each flexographic print unit 14, typically with hot air or ultraviolet light. Operator pushbuttons at the unit(s) 14 connect to the integrated PLC. Each action at one or more of the pushbuttons is evaluated by the integrated PLC in the host computer 40 which gets translated to machine commands, resulting in appropriate action by the motion controller(s) 44, 45.

The intaglio unit 17 generally follows the flexographic print unit(s) 14. The press will include one or more intaglio print units 17 and contain one or more inking units 23 each. Intaglio units 17 also include some form of wiping system 24 to remove excess ink applied to the plate. The intaglio unit(s) may also contain a prewipe system that removes excess ink from the plate. Each of the subsystems within the Intaglio unit(s) is controlled by the integrated PLC. Multiple servos coordinate the movement of paper through each intaglio unit.

When the press operator selects the operating mode of flexographic only, which may include other units on the press such as the aforementioned ones, the intaglio unit(s) 17 is inactive. The operator bypasses the intaglio unit(s) 17 by creating a web path around the unit(s). This is accomplished by passing the web 11 over rollers that route the web over or under said unit(s). Press motion commands from pushbuttons such as Jog and Run cause no movement within the Intaglio unit(s). Servo motor speed and position information is created using an internal synthesized method.

When combination print mode is selected, the first intaglio unit 17 in the press line generates the motion reference signal. All servo movement is coordinated by this signal. Flexographic unit(s)' servo(s) move in conjunction to the reference signal. The signal is conditioned by parametric data such as print repeat length, web tension set point, and the likes. Software algorithms continuously compute the associated servo commands which, when combined with servo motor digital position feedback, provide precise movement of the servo. It is said that this activity is referred to as an electronic gearbox. The software controls which signals and parameters are needed to effectively maintain registration between each dissimilar unit(s) in the press line. Complex algorithms message the reference signal before passing it on to the specific servo motors. The intaglio unit(s) utilize a "stop-n-go" substrate transport. This mechanism creates an



intermittent movement of the web similar to a movie projector. A series of four servos establish the intermittent motion. Since the Flexographic and other units in the press line require continuous and stable web movement, it is necessary for the "stop-n-go" transport to accept web as continuous flow and deliver web in a continuous flow to the next unit in the line. Each servo drives a pull roller 25 that the web contacts, thus allowing the servo to affect the position of the web. The implementation of the servos is as follows: 1) the first servo pulls web in a continuous manner from the previous unit. Its motion command from the motion controller is based on the reference signal and the parameters associated with web tension settings. It pushes the web into a vacuum box 26, which is used as an accumulator. A vacuum pump insures that adequate vacuum is present to receive web at the speed necessary. 2) The second servo, referred to as shuttle drive #1, pulls web from vacuum box #1 and passes it to the intaglio plate cylinder 27. The command for this servo is based on the reference signal and several other key elements. It is the responsibility of this servo to position the web in register with the image on the intaglio plate. The intaglio plate is mounted to a cylinder that has a circumference of double the largest Flexo plate cylinder. The intaglio plate cylinder is moving at twice the surface speed as the Flexo plate cylinder(s) so that both make a revolution in the same period of time. The shuttle #1 servo creates an intermittent motion pattern of web so that part of the time the web is stopped while part of the time the web is moving twice the speed as is the other units in the press line. The algorithm that generates the command to the servo uses the reference position to calculate the position of the shuttle 500 times per second. A mark sensor reads registration marks printed by the first unit in the press line to adjust the motion commands to permit perfect registration. It is the register mark that allows the motion controller to compensate for slippages in the web across rollers, slight elongations in the web, and other artifacts of the printing process. The motion algorithm uses a sine type pattern generator thereby creating a smooth transition from the "web stopped" portion of the cycle to "web moving" portion of the cycle. This sinusoidal pattern of motion is synchronized to the leading edge of the intaglio printing plate by the reference, and its relationship to the registration marks read on the web. Mark register data is collected through direct connection of the mark sensor to the motion controller board(s). 3) The third servo is referred to as shuttle servo #2. This servo accepts web from shuttle servo #1

after printed by the intaglio plate cylinder. The servo's command is identical to that of shuttle servo #1. with one exception: this servo command includes additional parametric data to create a slight difference in position relative to shuttle servo #1. In numerical terms, when shuttle servo #1 moves 1" forward, shuttle servo #2 may move 1.001" forward. This creates a slightly higher web tension while the web undergoes intaglio printing. Parametric data affects the amount of gain in shuttle servo #2. Shuttle servo #2 passes the web to vacuum box #2 (28), which serves as an accumulator as does vacuum box #1. 4) The fourth servo in the transport pulls web from vacuum box #2 and passes it to the next unit in the press line. This servo is running in a continuous motion mode, based on the reference signal. Parametric data establishes the electronic gear ratio so that the web is being delivered to the next unit at the same position and speed as the first servo accepted web from the previous unit on the press line.

Once the web 11 is printed with the intaglio ink, it passes through a forced air dryer 18. A chill unit 19 follows next in the press line, cooling the back to ambient temperature from a highly elevated temperature in the intaglio dryer 38. The web is passed across several chilled rollers 30 in a zigzag pattern. Circulating refrigerated water through the cores chills the rollers. The chilled rollers 30 are driven by a servo motor 35 that receives its command based on the reference and parametric data such as web tension settings. Additional intaglio unit(s) 17 would follow the first intaglio unit, if included. Their operation is identical to that of the first intaglio unit, described above.

As illustrated in FIG. 1B, finishing or processing units follow in the press line. These units may consist of one or more of the following: die cutting unit 33, registered holographic application unit, RFID applicator, numbering, or other. Each respective unit would be driven by a separate servo motor 35, which receives its command from the motion controller board(s). These units and their respective servo move the web in a continuous motion, using parametric data to affect registration and tension. Each unit may include a mark sensor to further affect the motion command. Precise registration is obtained when mark sensor data is part of the motion command, as web distortion, elongation, and the likes, become known in the calculations for the command. Likewise, conventional rewinding 37, folding, and/or sheeting apparatus may be included in the press line. Once the web is printed, and other features are added as necessary, the web

gets delivered as follows depending on customer requirements: A rewinder 37 accepts web from the previous unit in the press line and winds it onto paper cores. The size of the core and the overall dimension of the finished roll are dependent on the type of rewinder used and customer requirements. Typically, a roll would be wound on a 3 inch core and a 40 inch finish diameter. Rewinders are generally stand-alone units that receive basic run / stop information from the integrated PLC. Folders are driven by a servo motor that receives commands based on the reference and parametric data. A folder delivers the web in a fan folded format, and subsequently gets boxed in 2500 folded documents to a box. A sheeter is driven by a servo motor that receives commands from the motion controller(s) based on the reference signal and parametric data. A mark sensor normally accompanies the controls for a sheeter, as the cut position is a close registration feature. A sheeter accepts web from the previous unit in the press line and cuts the web into a equal length documents that are subsequently boxed by the 500 to 2500 unit count. The cut registration is accurately maintained when a mark sensor is incorporated to read the registration mark printed by the first active unit in the press line.

Software algorithms utilize digital data from devices on each of the units on the press line, incorporate that information with parametric data from the operator, and configuration data that specifies the resolution of each position encoder, the circumference of cylinders and pull rolls, and other machine specific data. It is this data that results in precise commands to each servo in the press that in turn results in accurate movement of the web under all operating conditions. Preferably, each servo has a separate algorithm evaluating the data and issuing commands. Electronic gearing can be thought of as the general activity of each algorithm. Additional computations take dynamics into account that include web stretch, web elongation, and other web distortions. In the case of the intaglio unit(s) 17, the shuttle roller 39 movement is based on a profile, or cam pattern. It's commanded position is continuously modified by position mark sensor data. The press operator adjusts parameters on the graphics display that results in web tension changes, as required by various substrate types. Algorithms adjust the servos commands so that the respective servo runs slightly slower or faster than the previous unit.

**Glossary:**

**Prewipe unit:** Used in conjunction with intaglio printing. An inking unit applies ink to the intaglio printing plate. The prewipe unit contacts the plate next for the purpose of removing excess ink from the printing plate. A typical configuration of a prewipe unit would include the prewipe roller, a doctor roller, and a doctor blade. The doctor roller removes the ink from the prewipe roller, and the doctor blade removes ink from the doctor roller. The removed ink is collected in a recovery system for either disposal or recycling.

**Reference:** This is the signal that orchestrates the movement of servos in the press line. It can be thought of in the same way as a conductor in an orchestra.

**Registration:** The alignment of multiple features to a substrate. An example is the alignment of two separate colors on a web. Another would be the alignment of printing on the web to that of a die cutting unit.

**Servo motor:** Refers to a high precision variable speed motor. Servos possess the ability to maintain precise speeds, accelerate or decelerate loads in a fraction of a second.

**Web:** The stream of paper or other substrate that spans the length of the press is the web. It starts at the unwinder, pulled from a roll, and extends to the delivery end of the press where it is either rewound into a roll, cut into sheets, or folded.

**Wiping System:** Part of an Intaglio print unit, the unit removes excess ink from the engraved printing plate. An intaglio inker applies ink to areas of a printing plate. The wiping system removes all ink from the surface of the plate thus leaving in only the engravings on a plate. When a prewipe system is included, it tends to reduce the load on the wiping system by taking a preliminary wipe on the plate. Wiping systems come in two forms, and are vastly different from each other. They are: 1) Water wipe system. A water wipe system utilizes a rubber covered roller that contacts the printing plate, rotating so that its surface is moving in the opposite direction as the plate cylinder. This effectively rubs the ink off the surface of the printing plate. Once the ink is collected on the wiping cylinder, it later comes in contact with cleaning solution, brushes, and a doctor blade. The process of wiping and cleaning is ongoing. The cleaning solution carries the excess ink away which is disposed in a separate process. 2) A paper wipe system removes ink similarly to the extent that the wiping roller is used to press paper against the plate, using

the same opposite direction movement. In this system, the ink is carried away on the paper, which starts at an unwinder, runs across the wiping roller, then rewound onto a paper core.

The press 10 utilizes three separate software programs (Appendix), each running on its own processor 40, 44, 45. An industrial PC 40 runs the host software with two RISC processor based motion control boards 44, 45 plugged into the industrial PC back plane. The industrial PC 40 acts as the host to coordinate the activities of itself and the motion control processors 44, 45. Operator interaction takes place through the host processor through the use of color graphic screen information and input through a keyboard/mouse combination or a touch screen. The motion control processors 44, 45 interpret commands from the host 40 via the PC back plane and carry out the activity of managing the machine motion throughout the press 10.

The first motion control processor 44 manages the activities of the intaglio unit 17 including the stop and go web transport. It also generates timing signals that one or more other motion control processors 45 utilize to synchronize web motion so as to maintain web tension and registration.

When the press 10 is operated in the combination mode where the intaglio printing is taking place in conjunction with the flexo printing and possibly other features on the press, the following process is utilized to manage motion: 1) The operator initiates a command to run the press via a run pushbutton; 2) the run command is interpreted by the host 40, and subsequently passes the command on to the first motion processor 44; 3) the first motion processor 44 enables the main drive 22 on the intaglio unit 17 and sends it a speed command; 4) the intaglio gear train, coupled with the main printing cylinder 27, rotates at the preset speed, which in turn causes the mechanically coupled reference encoder 21 to rotate; 5) the signal from the reference encoder is fed to the first motion processor 44; 6) the value of the reference encoder provides a binary number that points to a lookup table in the processor 44 signifying the position of each of the servo motors used to position the web 11; 7) the values in the lookup table are adjusted both statically when the press is calibrated and dynamically as a result of reading information about the web 11 on an ongoing basis via registration mark sensors and web tension sensors (the location of each of the sensors is dependant on the press configuration and

may or may not exist depending on the press configuration); 8) the first motion processor 44 also distributes the value of the reference encoder in real time across the PC back plane to the other motion processor(s) 45; 9) the other motion processor(s) 45 adjust the position of their respective servo motors based on their respective lookup tables.

When the press is operated without the intaglio unit 17, the first motion processor 44 generates a synthesized reference encoder signal that it uses and distributes to the other motion control board 45. In this case, the sequence of activity matches that of the combination mode with the following exceptions: 1) the host run and speed command is converted to the synthesized signal rather than generating the enable and speed commands to the intaglio main drive.

## Appendix

```

FrmProduction - 1
Option Explicit

Private Sub Command10_Click()
    Call ExitProduction
End Sub

Private Sub Command11_Click()
    'Get a login ID code from operator
    Dim I%
    WhoseFocus% = 4 'Tell Apad to come back to us
    ProductionTrace% = 2 'Return path
    frmApad!Text1.Text = ""
    frmApad.Visible = True

End Sub

Private Sub Command12_Click()
    'Handle the logout - simple at this point
    FrmProduction.Text1.Text = ""
    frmMain.Text2.Text = ""

End Sub

Private Sub Command13_Click()
    WhoseFocus% = 4 'This is the Production mode
    ProductionTrace% = 1 'Set return address
    frmNpad.Visible = True
    frmNpad!Text1.Text = "" 'Start with null

End Sub

Private Sub ExitProduction()
    FrmProduction.Visible = False
    frmMain.Text1.Text = "Standby"
    frmMain.SetFocus

End Sub

Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
    If KeyCode = &H79 Then Call ExitProduction

End Sub

Private Sub Text1_Change()
    'Sync Operator ID on frmMain with this variable
    frmMain!Text2.Text = FrmProduction!Text1.Text

End Sub

```



```

frmPressrun - 1
Option Explicit

Private Sub Command10_Click()

    ExitPressrun

End Sub

Private Sub Command11_Click()

    Dim I!
    I! = frmPressrun.MhRealInput3.Text - 0.005
    frmPressrun.MhRealInput3.Text = I!

End Sub

Private Sub Command12_Click()

    Dim I!
    I! = frmPressrun.MhRealInput3.Text + 0.005
    frmPressrun.MhRealInput3.Text = I!

End Sub

Private Sub ExitPressrun()

    frmPressrun.Visible = False
    Pressruntrace% = 0 'Clear return flag
    frmMain.Text1.Text = "Standby"
    frmMain.SetFocus

End Sub

Private Sub Command13_Click()

    Dim I!
    I! = frmPressrun.MhRealInput5.Text + 0.005
    If I! > (System_circumference# - 0.001) Then I! = 0
    frmPressrun.MhRealInput5.Text = I!

End Sub

Private Sub Command14_Click()

    Dim I!
    I! = frmPressrun.MhRealInput5.Text - 0.005
    If I! < 0 Then I! = System_circumference# - 0.005
    frmPressrun.MhRealInput5.Text = I!

End Sub

Private Sub Command15_Click()

    Dim I!
    I! = frmPressrun.MhRealInput3.Text - 0.1
    frmPressrun.MhRealInput3.Text = I!

End Sub

Private Sub Command16_Click()

    Dim I!
    I! = frmPressrun.MhRealInput3.Text + 0.1
    If I! > (System_circumference# - 0.001) Then I! = 0
    frmPressrun.MhRealInput3.Text = I!

```

```
frmPressrun - 2

End Sub

Private Sub Command17_Click()

    Dim I!
    I! = frmPressrun.MhRealInput5.Text - 0.1
    If I! < 0 Then I! = System_circumference# - 0.1
    frmPressrun.MhRealInput5.Text = I!

End Sub

Private Sub Command18_Click()

    Dim I!
    I! = frmPressrun.MhRealInput5.Text + 0.1
    If I! > (System_circumference# - 0.001) Then I! = 0
    frmPressrun.MhRealInput5.Text = I!

End Sub

Private Sub Command19_Click()

    'Toggle Intaglio Auto/Manual
    If RegIntaglioAuto% = 0 Then
        RegIntaglioAuto% = 1
        frmPressrun!Label32.Visible = True
    Else
        RegIntaglioAuto% = 0
        frmPressrun!Label32.Visible = False
    End If
    VarChangeFlag%(5) = 1

End Sub

Private Sub Command20_Click()

    'Toggle Cutoff Auto/Manual
    If RegCutoffAuto% = 0 Then
        RegCutoffAuto% = 1
        frmPressrun!Label4.Visible = True
    Else
        RegCutoffAuto% = 0
        frmPressrun!Label4.Visible = False
    End If
    VarChangeFlag%(8) = 1

End Sub

Private Sub Command21_Click()

    Dim I!
    I! = frmPressrun.MhRealInput2.Text + 0.1
    If I! > (System_circumference# - 0.001) Then I! = 0
    frmPressrun.MhRealInput2.Text = I!

End Sub

Private Sub Command22_Click()

    Dim I!
    I! = frmPressrun.MhRealInput2.Text - 0.1
    If I! < 0 Then I! = System_circumference# - 0.1
    frmPressrun.MhRealInput2.Text = I!

End Sub

Private Sub Command23_Click()
```

```
frmPressrun - 3
```

```
Dim I!
I! = frmPressrun.MhRealInput2.Text + 0.005
If I! > (System_circumference# - 0.001) Then I! = 0
frmPressrun.MhRealInput2.Text = I!
```

```
End Sub
```

```
Private Sub Command24_Click()
```

```
Dim I!
I! = frmPressrun.MhRealInput2.Text - 0.005
If I! < 0 Then I! = System_circumference# - 0.005
frmPressrun.MhRealInput2.Text = I!
```

```
End Sub
```

```
Private Sub Command25_Click()
```

```
'Toggle Numbering Auto/Manual
If RegNumberingAuto% = 0 Then
    RegNumberingAuto% = 1
    frmPressrun!Label7.Visible = True
Else
    RegNumberingAuto% = 0
    frmPressrun!Label7.Visible = False
End If
VarChangeFlag%(7) = 1
```

```
End Sub
```

```
Private Sub Command26_Click()
```

```
frmPressrun!Slider1.Value = frmPressrun!Slider1.Value + 10
```

```
End Sub
```

```
Private Sub Command27_Click()
```

```
'Toggle Diecutting Auto/Manual
If RegDiecuttingAuto% = 0 Then
    RegDiecuttingAuto% = 1
    frmPressrun!Label28.Visible = True
Else
    RegDiecuttingAuto% = 0
    frmPressrun!Label28.Visible = False
End If
VarChangeFlag%(6) = 1
```

```
End Sub
```

```
Private Sub Command28_Click()
```

```
Dim I!
I! = frmPressrun.MhRealInput6.Text + 0.1
frmPressrun.MhRealInput6.Text = I!
```

```
End Sub
```

```
Private Sub Command29_Click()
```

```
Dim I!
I! = frmPressrun.MhRealInput6.Text - 0.1
frmPressrun.MhRealInput6.Text = I!
```

```
End Sub
```

```

frmPressrun - 4

Private Sub Command30_Click()

    Dim I!
    I! = frmPressrun.MhRealInput6.Text! + 0.005
    frmPressrun.MhRealInput6.Text! = I!

End Sub

Private Sub Command31_Click()

    Dim I!
    I! = frmPressrun.MhRealInput6.Text! - 0.005
    frmPressrun.MhRealInput6.Text! = I!

End Sub

Private Sub Command32_Click()

    'Scrap Blower Control
    If ScrapRequest# = 0 Then
        ScrapRequest# = 1
        frmPressrun!Label30.Visible = True
    Else
        ScrapRequest# = 0
        frmPressrun!Label30.Visible = False
    End If

End Sub

Private Sub Command33_Click()

    'Numbering 1 Control
    Call Numbers1_Man_Toggle

End Sub

Private Sub Command34_Click()

    frmPressrun!Slider1.Value = frmPressrun!Slider1.Value - 10

End Sub

Private Sub Command35_Click()

    'Numbering 2 Control
    Call Numbers2_Man_Toggle

End Sub

Private Sub Command37_Click()

    Dim I!
    I! = frmPressrun.MhRealInput4.Text! + 0.1
    If I! > (System_circumference# - 0.001) Then I! = 0
    frmPressrun.MhRealInput4.Text! = I!

End Sub

Private Sub Command38_Click()

    Dim I!
    I! = frmPressrun.MhRealInput4.Text! - 0.1
    If I! < 0 Then I! = System_circumference# - 0.1
    frmPressrun.MhRealInput4.Text! = I!

End Sub

```

```
frmPressrun = 5
```

```
Private Sub Command33_Click()
```

```
Dim I!
T1 = frmPressrun.MhRealInput1.Text1 + 0.005
If I! > (System.Environment# 0.001) Then I! = 0
frmPressrun.MhRealInput1.Text1 = T1
```

```
End Sub
```

```
Private Sub Command34_Click()
```

```
Dim I!
T1 = frmPressrun.MhRealInput1.Text1 - 0.005
If I! < 0 Then I! = System.Environment# 0.001
frmPressrun.MhRealInput1.Text1 = T1
```

```
End Sub
```

```
Private Sub Command7_Click()
```

```
WhoseFocus# = 1 'This is calling module TD
PressrunTrace# 4 'Set return address
frmNpad.Visible = True
frmNpad1Text11.Text1 = "" 'Start with null
```

```
End Sub
```

```
Private Sub Command8_Click()
```

```
WhoseFocus# = 1 'This is calling module TD
PressrunTrace# 3 'Set return address
frmNpad.Visible = True
frmNpad1Text11.Text1 = "" 'Start with null
```

```
End Sub
```

```
Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
```

```
If KeyCode = &H76 Then
    WhoseFocus# = 3 'This is calling module TD
    PressrunTrace# 2 'Set return address
    frmNpad.Visible = True
    frmNpad1Text11.Text1 = "" 'Start with null
End If
If KeyCode = &H75 Then Call ExitPressrun
```

```
End Sub
```

```
Private Sub MhCmdF101_Click()
```

```
End Sub
```

```
Private Sub MhRealInput1_Click(Index As Integer)
```

```
WhoseFocus# 1 'Caller ID
PressrunTrace# Index# + 5 'Variable ID
frmNpad.Visible = True
frmNpad1Text11.Text1 = "" 'Clear buffer
```

```
End Sub
```

```
Private Sub MhRealInput2_Change()
```

```
'Numbering register offset changed
```

```
frmPressorun = 6
```

```
VarChangeFlag(2) = 1
```

```
End Sub
```

```
Private Sub MRCallInput2_Click()
```

```
WhoseFocus% = 1 'Caller ID  
frmXunitracem% = 2 'Numbering register offset.  
frmNpad.Visible = True  
frmNpad!Text1.Text = "" 'Clear buffer
```

```
End Sub
```

```
Private Sub MRCallInput3_Change()
```

```
'Tnlag10 register offset changed  
VarChangeFlag(0) = 1
```

```
End Sub
```

```
Private Sub MRCallInput3_Click()
```

```
WhoseFocus% = 1 'Caller ID  
frmXunitracem% = 1 'Tnlag10 register offset.  
frmNpad.Visible = True  
frmNpad!Text1.Text = "" 'Clear buffer
```

```
End Sub
```

```
Private Sub MRCallInput4_Change()
```

```
'Flaxo register offset changed  
VarChangeFlag(45) = 1
```

```
End Sub
```

```
Private Sub MRCallInput5_Click()
```

```
WhoseFocus% = 1 'Caller ID  
frmXunitracem% = 1 'Flaxo register offset.  
frmNpad.Visible = True  
frmNpad!Text1.Text = "" 'Clear buffer
```

```
End Sub
```

```
Private Sub MRCallInput6_Change()
```

```
'CuloFF register offset changed  
VarChangeFlag(3) = 1
```

```
End Sub
```

```
Private Sub MRCallInput6_Click()
```

```
WhoseFocus% = 1 'Caller ID  
frmXunitracem% = 3 'CuloFF register offset.  
frmNpad.Visible = True  
frmNpad!Text1.Text = "" 'Clear buffer
```

```
End Sub
```

```
Private Sub MRCallInput6b_Change()
```

```
'Disemling register offset changed  
VarChangeFlag(1) = 1
```

```
End Sub
```

```
frmPressrun 7
```

```
Private Sub HbRealignOut_Click()
```

```
    WhoseFocus = 1 'Caller ID
    PressrunTrace = 17 'Disabling register offset
    frmHb.Visible = True
    frmHb.Text1.Text = "" 'Clear buffer
```

```
End Sub
```

```
Private Sub Slider1_Change()
```

```
    frmPressrun.Text2.Text = frmPressrun.Slider1.Value
    frmPUFInfo.Text2.Text = frmPressrun.Text2.Text
    frmPUFInfo.Slider1.Value = frmPressrun.Slider1.Value
```

```
End Sub
```

```
Private Sub Text2_Change()
```

```
    'Update other screens spin/speed fields
    frmIStatus.Text1.Text = frmPressrun.Text2.Text
    frmPUFInfo.Text1.Text = frmPressrun.Text2.Text
    TargetSpinSpeed = frmPressrun.Text2.Text
    VmChangeFlag(5) = 1
```

```
End Sub
```

```
Private Sub Text2_Click()
```

```
    WhoseFocus = 1 'This is calling module to
    PressrunTrace = 1 'Set return address
    frmHb.Visible = True
    frmHb.Text1.Text = "" 'Clear with null
```

```
End Sub
```

```

frmP00Intaglio - 1
Option Explicit
Private Sub Command10_Click()
    ExitIntaglioSetup
End Sub

Private Sub Command13_Click()
    'Set Wiping rewinder to manual
    WipingWinderEna% = 1
    frmP00Intaglio!Label57.Visible = False
    frmP00Intaglio!Label7.Visible = True
    frmP00Intaglio!Label5.Visible = False
End Sub

Private Sub Command14_Click()
    'Turn hydraulics to low
    frmP00Intaglio!Label52.Visible = True
    frmP00Intaglio!Label51.Visible = False
    frmP00Intaglio!Label53.Visible = False
    Hydraulics% = 1
End Sub

Private Sub Command18_Click()
    'Turn hydraulics off
    frmP00Intaglio!Label53.Visible = True
    frmP00Intaglio!Label51.Visible = False
    frmP00Intaglio!Label52.Visible = False
    Hydraulics% = 2
End Sub

Private Sub ExitIntaglioSetup()
    frmP00Intaglio.Visible = False
    P00Intagliotrace% = 0 'Clear return flag
    frmMain.Text1.Text = "Standby"
    frmMain.SetFocus
End Sub

Private Sub Command30_Click()
    Call Hyd_Init_Off
End Sub

Private Sub Command11_Click()
    'Intaglio Inker #1 mode: Manual
    If MachineRunning% = 0 Then
        IntaglioInker1Control% = 1
        frmP00Intaglio!Label43.Visible = True
        frmP00Intaglio!Label42.Visible = False
        frmP00Intaglio!Label44.Visible = False
        frmP00Intaglio!Command12.Enabled = True
        frmP00Intaglio!Command32.Enabled = True
        VarChangeFlag%(34) = 1
    End If

```



```

frmP00Intaglio - 2

End Sub

Private Sub Command12_Click()
    Call InkerIrun_Man_Toggle
End Sub

Private Sub Command15_Click()
    'Intaglio Inker #1 mode: Auto
    If MachineRunning% = 0 Then
        IntaglioInker1Control% = 2
        frmP00Intaglio!Label144.Visible = True
        frmP00Intaglio!Label143.Visible = False
        frmP00Intaglio!Label142.Visible = False
        VarChangeFlag%(34) = 1
        IntaglioInker1nip% = 0
        IntaglioInker1Manrun% = 0
        frmP00Intaglio!Label126.Visible = False
        frmP00Intaglio!Label127.Visible = False
        frmP00Intaglio!Command12.Enabled = False
        frmP00Intaglio!Command32.Enabled = False
    End If
End Sub

Private Sub Command16_Click()
    Call IntInker1_Init_Off
End Sub

Private Sub Command17_Click()
    Call IntImps_Init_Off
End Sub

Private Sub Command19_Click()
    'Intaglio Impression mode: Auto
    If MachineRunning% = 0 Then
        IntaglioImpsControl% = 2
        frmP00Intaglio!Label16.Visible = True
        frmP00Intaglio!Label18.Visible = False
        frmP00Intaglio!Label19.Visible = False
        VarChangeFlag%(36) = 1
        If IntaglioImpsManual% <> 0 Then
            IntaglioImpsManual% = 0
            frmP00Intaglio!Label119.Visible = False
        End If
        frmP00Intaglio!Command22.Enabled = False
    End If
End Sub

Private Sub Command20_Click()
    'Intaglio Impression mode: Manual
    If MachineRunning% = 0 Then
        IntaglioImpsControl% = 1
        frmP00Intaglio!Label18.Visible = True
        frmP00Intaglio!Label19.Visible = False
        frmP00Intaglio!Label16.Visible = False
        frmP00Intaglio!Command22.Enabled = True
        VarChangeFlag%(36) = 1
    End If
End Sub

```

```

frmP00Intaglio - 3

End If

End Sub

Private Sub Command21_Click()

    Call IntWipe_Init_Off

End Sub

Private Sub Command22_Click()

    Call IntImps_Man_Toggle

End Sub

Private Sub Command23_Click()

    'Intaglio Wiping mode: Auto
    If MachineRunning% = 0 Then
        IntaglioWipeControl% = 2
        frmP00Intaglio!Label13.Visible = True
        frmP00Intaglio!Label14.Visible = False
        frmP00Intaglio!Label15.Visible = False
        VarChangeFlag%(37) = 1
        If IntaglioWipeManual% <> 0 Then
            IntaglioWipeManual% = 0
            frmP00Intaglio!Label16.Visible = False
        End If
        frmP00Intaglio!Command25.Enabled = False
    End If

End Sub

Private Sub Command24_Click()

    'Intaglio Wiping mode: Manual
    If MachineRunning% = 0 Then
        IntaglioWipeControl% = 1
        frmP00Intaglio!Label14.Visible = True
        frmP00Intaglio!Label15.Visible = False
        frmP00Intaglio!Label13.Visible = False
        frmP00Intaglio!Command25.Enabled = True
        VarChangeFlag%(37) = 1
    End If

End Sub

Private Sub Command25_Click()

    Call IntWipe_Man_Toggle

End Sub

Private Sub Command26_Click()

    Call IntPrewipe_Init_Off

End Sub

Private Sub Command27_Click()

    'Intaglio Prewipe mode: Auto
    If MachineRunning% = 0 Then
        IntaglioPrewipeControl% = 2
        frmP00Intaglio!Label18.Visible = True
        frmP00Intaglio!Label20.Visible = False
    End If

End Sub

```

```

frmP00Intaglio - 4

    frmP00Intaglio!Label21.Visible = False
    VarChangeFlag%(38) = 1
    If IntaglioPrewipeManual% <> 0 Then
        IntaglioPrewipeManual% = 0
        frmP00Intaglio!Label22.Visible = False
    End If
    frmP00Intaglio!Command29.Enabled = False
End If

End Sub

Private Sub Command28_Click()

    'Intaglio Prewipe mode: Manual
    If MachineRunning% = 0 Then
        IntaglioPrewipeControl% = 1
        frmP00Intaglio!Label20.Visible = True
        frmP00Intaglio!Label21.Visible = False
        frmP00Intaglio!Label18.Visible = False
        frmP00Intaglio!Command29.Enabled = True
        VarChangeFlag%(38) = 1
    End If

End Sub

Private Sub Command29_Click()

    Call IntPrewipe_Man_Toggle

End Sub

Private Sub Command31_Click()

    'Set Wiping rewiner to auto
    WipingWinderEna% = 2
    frmP00Intaglio!Label57.Visible = False
    frmP00Intaglio!Label17.Visible = False
    frmP00Intaglio!Label15.Visible = True

End Sub

Private Sub Command32_Click()

    Call Inker1nip_Man_Toggle

End Sub

Private Sub Command33_Click()

    Call IntInker2_Init_Off

End Sub

Private Sub Command34_Click()

    'Intaglio Inker #2 mode: Auto
    If MachineRunning% = 0 Then
        IntaglioInker2Control% = 2
        frmP00Intaglio!Label32.Visible = True
        frmP00Intaglio!Label33.Visible = False
        frmP00Intaglio!Label34.Visible = False
        VarChangeFlag%(35) = 1
        IntaglioInker2nip% = 0
        IntaglioInker2Manrun% = 0
        frmP00Intaglio!Label41.Visible = False
        frmP00Intaglio!Label45.Visible = False
    End If

```

```

frmP00TnlAglio = 5
    frmP00TnlAglio!Command35.Enabled = False
    frmP00TnlAglio!Command37.Enabled = False
End If

End Sub

Private Sub Command36_Click()
    'TnlAglio Take #2 move Manual
    If MachineRunning = 0 Then
        TnlAglioTake2Control = 1
        frmP00TnlAglio!Label33.Visible = True
        frmP00TnlAglio!Label34.Visible = False
        frmP00TnlAglio!Label35.Visible = False
        frmP00TnlAglio!Command36.Enabled = True
        frmP00TnlAglio!Command37.Enabled = True
        VarChangeFlag8(35) = 1
    End If
End Sub

Private Sub Command36_Click()
    Call Inker2run_Man_Toggle
End Sub

Private Sub Command37_Click()
    Call Inker2nip_Man_Toggle
End Sub

Private Sub Command38_Click()
    'Set Wiping rewinder to off
    WipingWinderEna = 0
    frmP00TnlAglio!Label57.Visible = True
    frmP00TnlAglio!Label17.Visible = False
    frmP00TnlAglio!Label15.Visible = False
End Sub

Private Sub Command6_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    ProductCalReq8 = True
    VarChangeFlag8(24) = 1
End Sub

Private Sub Command6_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
    ProductCalReq8 = False
    VarChangeFlag8(24) = 1
End Sub

Private Sub Command7_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    WipingCalReq8 = True
    VarChangeFlag8(25) = 1
End Sub

Private Sub Command7_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
    WipingCalReq8 = False

```

```

frmP00Tntaglio = A
    VarChangeFlag%(24) = 1
End Sub

Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
    If KeyCode = &H75 Then Call ExitTntaglioSetup
End Sub

Private Sub MhRealInput3_Click()
    WhoseFocus% = 'Caller ID
    P00TntaglioInFocus% = 4 'Press repeat button
    frmP00Tntaglio.Visible = True
    frmP00Tntaglio!Text1.Text = "" 'Clear buffer
End Sub

Private Sub MhRealInput4_Click()
    WhoseFocus% = 'Caller ID
    P00TntaglioInFocus% = 5 'Press repeat button
    frmP00Tntaglio.Visible = True
    frmP00Tntaglio!Text1.Text = "" 'Clear buffer
End Sub

Private Sub label154_Click()
End Sub

Private Sub Slider1_Change()
    'Manual Speed value Input #1
    frmP00Tntaglio!MhRealInput2.Text = frmP00Tntaglio!Slider1.Value
    VarChangeFlag%(31) = 1
End Sub

Private Sub Slider2_Change()
    'Auto Speed trim Input #1
    frmP00Tntaglio!MhRealInput1.Text = frmP00Tntaglio!Slider2.Value
    VarChangeFlag%(30) = 1
End Sub

Private Sub Slider3_Change()
    'Auto Speed trim Input #2
    frmP00Tntaglio!MhRealInput3.Text = frmP00Tntaglio!Slider3.Value
    VarChangeFlag%(32) = 1
End Sub

Private Sub Slider4_Change()
    'Manual Speed value Input #2
    frmP00Tntaglio!MhRealInput4.Text = frmP00Tntaglio!Slider4.Value
    VarChangeFlag%(33) = 1
End Sub

Private Sub Slider5_Change()

```

1. Introduction

Writings of the author in various languages

1.  $\text{C}_{10}\text{H}_8$  2.  $\text{C}_{10}\text{H}_8$  3.  $\text{C}_{10}\text{H}_8$  4.  $\text{C}_{10}\text{H}_8$  5.  $\text{C}_{10}\text{H}_8$  6.  $\text{C}_{10}\text{H}_8$  7.  $\text{C}_{10}\text{H}_8$  8.  $\text{C}_{10}\text{H}_8$  9.  $\text{C}_{10}\text{H}_8$  10.  $\text{C}_{10}\text{H}_8$

ಪುನಃ ಪರಿಶೀಲನೆ

```

frmP00Flexo . 1
Option Explicit
Private Sub Command1_Click()
    Call Newfile
End Sub
Private Sub Command10_Click()
    ExitP00Flexo
End Sub
Private Sub Command11_Click()
    'Flexo Impression mode: Manual
    If MachineRunning% = 0 Then
        FlexoImpsControl% = 1
        frmP00Flexo!Label2.Visible = True
        frmP00Flexo!Label11.Visible = False
        frmP00Flexo!Label6.Visible = False
        frmP00Flexo!Command22.Enabled = True
        VarChangeFlag%(39) = 1
    End If
End Sub
Private Sub Command12_Click()
    'Flexo Impression mode: Auto
    If MachineRunning% = 0 Then
        FlexoImpsControl% = 2
        frmP00Flexo!Label6.Visible = True
        frmP00Flexo!Label11.Visible = False
        frmP00Flexo!Label2.Visible = False
        VarChangeFlag%(39) = 1
        If FlexoImpsManual% <> 0 Then
            FlexoImpsManual% = 0
            frmP00Flexo!Label19.Visible = False
        End If
        frmP00Flexo!Command22.Enabled = False
    End If
End Sub
Private Sub Command13_Click()
    'Flexo Impression mode: Off
    If MachineRunning% = 0 Then
        FlexoImpsControl% = 0
        frmP00Flexo!Label11.Visible = True
        frmP00Flexo!Label2.Visible = False
        frmP00Flexo!Label6.Visible = False
        VarChangeFlag%(39) = 1
        If FlexoImpsManual% <> 0 Then
            FlexoImpsManual% = 0
            frmP00Flexo!Label19.Visible = False
        End If
        frmP00Flexo!Command22.Enabled = False
    End If
End Sub
Private Sub Command14_Click()

```

```

frmF00Flexo 2

'Toggle the Cutoff Unit Enable
If MachineRunning% = 0 Then
    If CutoffEnable% = 0 Then 'We're offline
        frmF00Flexo!Label16.BackColor = &H00FF00
        CutoffEnable% = 1
    Else
        frmF00Flexo!Label16.BackColor = &H00FFFF
        CutoffEnable% = 0
    End If
    VarChangeFlag%(28) = 1
End If

End Sub

Private Sub Command17_Click()

'Toggle the guard bypass switch
If GuardsBypassed% = 0 Then
    GuardsBypassed% = 1
    frmF00Flexo!Label36.Visible = True
Else
    GuardsBypassed% = 0
    frmF00Flexo!Label36.Visible = False
End If

End Sub

Private Sub Command18_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)

'Clear ALL calibration flags
MachineCalibrated% = 0
frmF00Flexo!Label37.Visible = True

End Sub

Private Sub Command18_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)

'Turn off light
frmF00Flexo!Label37.Visible = False

End Sub

Private Sub Command19_Click()

'Set System Mode to Combination mode
If MachineRunning% = 0 Then
    SysMode% = 2
    IntInfeedEnable% = 1
    IntShuttleEnable% = 1
    IntOutfeedEnable% = 1
    IntChillrollEnable% = 1
    IntWipingEnable% = 1
    VarChangeFlag%(4) = 1
    frmF00Flexo!Label40.Visible = True
    frmF00Flexo!Label41.Visible = False
    frmF00Flexo!Label42.Visible = False
End If

End Sub

Private Sub Command2_Click()

Call Openfile

End Sub

Private Sub Command20_Click()

```



frmP00Flexo 3

```

'Set System Mode to Flexo only
If MachineRunning% = 0 Then
    SysMode% = 0
    IntInfeedEnable% = 0
    IntShuttleEnable% = 0
    IntOutfeedEnable% = 0
    IntChillrollEnable% = 0
    IntWipingEnable% = 0
    VarChangeFlag%(4) = 1
    frmP00Flexo!Label42.Visible = True
    frmP00Flexo!Label41.Visible = False
    frmP00Flexo!Label40.Visible = False
End If

```

End Sub

Private Sub Command21\_Click()

```

'Set System Mode to Intaglio only
If MachineRunning% = 0 Then
    SysMode% = 1
    IntInfeedEnable% = 1
    IntShuttleEnable% = 1
    IntOutfeedEnable% = 1
    IntChillrollEnable% = 1
    IntWipingEnable% = 1
    VarChangeFlag%(4) = 1
    frmP00Flexo!Label41.Visible = True
    frmP00Flexo!Label42.Visible = False
    frmP00Flexo!Label40.Visible = False
End If

```

End Sub

Private Sub Command22\_Click()

```

'Flexo Imps manual control
Call FlexoImps_Man_Toggle

```

End Sub

Private Sub Command25\_Click()

```

'Toggle the Numbering Unit Enable
If MachineRunning% = 0 Then
    If NumberingEnable% = 0 Then 'We're offline
        frmP00Flexo!Label13.BackColor = &H00FF00
        NumberingEnable% = 1
    Else
        frmP00Flexo!Label13.BackColor = &H90FF9F
        NumberingEnable% = 0
    End If
    VarChangeFlag%(27) = 1
End If

```

End Sub

Private Sub Command26\_Click()

```

frmPressrun!Slider1.Value = frmPressrun!Slider1.Value + 10
frmP00Flexo!Slider1.Value = frmPressrun!Slider1.Value

```

End Sub

Private Sub Command27\_Click()

```

frmPOOFlexo 4

'Toggle the Decutting Unit Enable
If MachineRunning? = 0 Then
    If DecuttingEnable? = 0 Then 'We're offline
        frmPOOFlexo!Label13.BackColor = &H00FF00
        DecuttingEnable? = 1
    Else
        frmPOOFlexo!Label13.BackColor = &H00FFFF
        DecuttingEnable? = 0
    End If
    VsrChangeFlag?(26) = 1
End If

End Sub

Private Sub Command3_Click()

    Call SaveFile

End Sub

Private Sub Command34_Click()

    frmPressrun!Slider1.Value = frmPressrun!Slider1.Value + 10
    frmPOOFlexo!Slider1.Value = frmPressrun!Slider1.Value

End Sub

Private Sub Command4_Click()

    Call SaveAsFile

End Sub

Private Sub ExitPOOFlexo()

    Dim I%

    POOFlexotrace? = 0 'Clear return path
    frmPOOFlexo.Visible = False
    frmMain!Text1.Text = "Standby"

End Sub

Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)

    If KeyCode = &H70 Then Call Newfile
    If KeyCode = &H71 Then Call Openfile
    If KeyCode = &H72 Then Call SaveFile
    If KeyCode = &H73 Then Call SaveAsFile
    If KeyCode = &H75 Then Call ExitPOOFlexo

End Sub

Private Sub Newfile()

    'This procedure initializes all variables for a new
    'job.

    Dim I%

    WhoseFocus? = 2 'Tell Apad to come back to us
    POOFlexotrace? = 1 'Return path

    frmApad!Text1.Text = ""
    frmApad.Visible = True

End Sub

```

```
DimFOCFocus% = 0
```

```
Private Sub OpenFile()
```

```
    'This procedure loads an existing job profile from disk
```

```
    Dim I%
```

```
    WhoseFocus% = 2 'Tell Apad to come back to us  
    POCFlxolracc% = 2 'Return path
```

```
    frmApad!Text11.Text = ""  
    frmApad.Visible = True
```

```
End Sub
```

```
Private Sub SaveAsFile()
```

```
    'This procedure saves the current job profile data to a  
    'user defined filename.
```

```
    Dim I%
```

```
    WhoseFocus% = 2 'Tell Apad to come back to us  
    POCFlxolracc% = 4 'Return path
```

```
    frmApad!Text11.Text = ""  
    frmApad.Visible = True
```

```
End Sub
```

```
Private Sub SaveFile()
```

```
    'This procedure saves the current job profile to disk
```

```
    Dim I%
```

```
    I% = SaveJobdata()
```

```
End Sub
```

```
Private Sub MhRealInput5_Change()
```

```
End Sub
```

```
Private Sub MhRealInput5_Click()
```

```
    WhoseFocus% = 2 'Caller ID  
    POCFlxolracc% = 1 'Press repeat length  
    frmApad.Visible = True  
    frmApad!Text11.Text = "" 'Clear buffer
```

```
End Sub
```

```
Private Sub MhRealInput6_Change()
```

```
End Sub
```

```
Private Sub MhRealInput6_Click()
```

```
    WhoseFocus% = 2 'Caller ID  
    POCFlxolracc% = 2 'Inlaglio repeat factor  
    frmApad.Visible = True  
    frmApad!Text11.Text = "" 'Clear buffer
```

```
End Sub
```

```
Private Sub Slider1_Change()
```

```
frmPOUFlexo = 6

    frmPressrun!Slider1.Value = frmPOUFlexo!Slider1.Value
End Sub

Private Sub Text3_Change()
    'Update other screens speed field

    frmPressrun!Text3.Text = frmPOUFlexo!Text3.Text
    frmIstatus!Text3.Text = frmPOUFlexo!Text3.Text
    frmPOUInaglio!Text3.Text = frmPOUFlexo!Text3.Text
End Sub
```

```

frmMain 1

Option Explicit

Private Sub BeginIstatus()

    frmMain!Text1.Text = "Interlocks"

    FrmIstatus.Top = frmMain.Top + 1890
    FrmIstatus.Left = frmMain.Left + 30
    FrmIstatus.Visible = True

End Sub

Private Sub BeginPressRunning()

    frmMain!Text1.Text = "Run statistics"

    frmPressrun.Top = frmMain.Top + 1890
    frmPressrun.Left = frmMain.Left + 30
    frmPressrun.Visible = True

End Sub

Private Sub BeginParameter()

    If (SystemStatus? And 9) = 0 Then 'Not if we're in RUN or Pause
        frmParam.Top = frmMain.Top + 9925
        frmParam.Left = frmMain.Left + 30
        frmParam.Visible = True
        frmPassW.Visible = True
    End If

End Sub

Private Sub BeginProduction()

    If Param_tbl$(991, 2) = "0" Then

        FrmProduction.Top = frmMain.Top + 1890
        FrmProduction.Left = frmMain.Left + 30
        FrmProduction.Visible = True

        frmMain!Text1.Text = "Production Data"
    End If

End Sub

Private Sub BeginSetupFlexo()

    If Param_tbl$(989, 2) = "0" Then

        frmMain!Text1.Text = "Flexo setup"

        frmP00Flexo.Top = frmMain.Top + 1890
        frmP00Flexo.Left = frmMain.Left + 30
        frmP00Flexo.Visible = True

    End If

End Sub

Private Sub BeginSetupIntaglio()

    If Param_tbl$(989, 2) = "0" Then

        frmMain!Text1.Text = "Intaglio setup"

        frmP00Intaglio.Top = frmMain.Top + 1890
        frmP00Intaglio.Left = frmMain.Left + 30
    End If

End Sub

```

```
frmMain - 2
    frmP00Intaglio.Visible = True
End If
End Sub

Private Sub BeginUtilities()
    frmMain!Text1.Text = "Utilities"
    frmUtils.Top = frmMain.Top + 1890
    frmUtils.Left = frmMain.Left + 30
    frmUtils.Visible = True
End Sub

Private Sub Command1_Click()
    BeginPressRunning
End Sub

Private Sub Command10_Click()
    BeginParameter
End Sub

Private Sub Command2_Click()
    BeginProduction
End Sub

Private Sub Command3_Click()
    BeginSetupIntaglio
End Sub

Private Sub Command4_Click()
    BeginSetupFlexo
End Sub

Private Sub Command5_Click()
    BeginIstatus
End Sub

Private Sub Command9_Click()
    BeginUtilities
End Sub

Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
    If KeyCode = &H70 Then BeginPressRunning
    If KeyCode = &H71 Then BeginProduction
    If KeyCode = &H72 Then BeginSetupIntaglio
    If KeyCode = &H73 Then BeginSetupFlexo
    If KeyCode = &H74 Then BeginIstatus
    If KeyCode = &H78 Then BeginUtilities
    If KeyCode = &H79 Then BeginParameter
```

```

frmMain - 3

End Sub

Private Sub Form_Load()
    Call Init_System
    Call PLC_FirstScan 'Initialize PLC registers
End Sub

Private Sub Timer1_Timer()
    'This is the motion delay timer
    'Used to warn operator of press movement - 2 seconds

    Startdelay% = 2 'Set the start delay flag
    frmMain.Timer1.Enabled = False
End Sub

Private Sub Timer2_Timer()
    Dim I%

    I% = SysPowerCheck%() 'Test Servo power status
    I% = AutoExec%() 'Call Master System RUN processor
End Sub

Private Sub Timer3_Timer()
    'This is the one second update calls; primarily updating
    'graphics on the operator console.

    Call Update_production_screen
    Call Update_pressrun_screen
End Sub

Private Sub Timer4_Timer()
    'Long term updates to Disk, etc. happen here
    'Timer is set for one minute updates

    Call WriteCounters
    Call SaveRolldata
End Sub

Private Sub Timer5_Timer()
    'This timer calls the PLC Scan routines. Loop time needs to be
    'in the neighborhood of 10mS.

    Call PLC_Scan
End Sub

Private Sub Timer6_Timer()
    'This timer is used to create a ride-thru time. It's the time that
    'motion can be re-initiated after a stop.

    Ridethrudelay% = 0 'Clear the wait flag
    frmMain.Timer6.Enabled = False

```

frmMain - 4

End Sub



```

FrmIstatus - 1
Option Explicit
Private Sub Command10_Click()
    Call ExitIstatus
End Sub
Private Sub ExitIstatus()
    FrmIstatus.Visible = False
    frmMain.Text1.Text = "Standby"
    frmMain.SetFocus
End Sub
Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
    If KeyCode = &H79 Then Call ExitIstatus
End Sub
Private Sub Timer1_Timer()
    Call UpdateIlockDisplay
End Sub
Public Sub UpdateIlockDisplay()
    'This procedure reads the appropriate status information and
    'updates the display. Timer set to refresh once per second.

    If PLC_I$(48) = 0 Then 'Unwinder Web break status
        FrmIstatus!Mh3dLabel1.FillColor = &HFF 'Red
    Else
        FrmIstatus!Mh3dLabel1.FillColor = &H80FF00 'Green
    End If

    If PLC_I$(49) = 0 Then 'Flexo Web break status
        FrmIstatus!Mh3dLabel2.FillColor = &HFF 'Red
    Else
        FrmIstatus!Mh3dLabel2.FillColor = &H80FF00 'Green
    End If

    If PLC_I$(50) = 0 Then 'Intaglio infeed Web break status
        FrmIstatus!Mh3dLabel3.FillColor = &HFF 'Red
    Else
        FrmIstatus!Mh3dLabel3.FillColor = &H80FF00 'Green
    End If

    If PLC_I$(17) = 0 Then 'Intaglio outfeed Web break status
        FrmIstatus!Mh3dLabel4.FillColor = &HFF 'Red
    Else
        FrmIstatus!Mh3dLabel4.FillColor = &H80FF00 'Green
    End If

    If PLC_I$(51) = 0 Then 'Chill Web break status
        FrmIstatus!Mh3dLabel5.FillColor = &HFF 'Red
    Else
        FrmIstatus!Mh3dLabel5.FillColor = &H80FF00 'Green
    End If

    If PLC_I$(52) = 0 Then 'Diecut Web break status
        FrmIstatus!Mh3dLabel6.FillColor = &HFF 'Red
    Else
        FrmIstatus!Mh3dLabel6.FillColor = &H80FF00 'Green
    End If

```

FrmIstatus - 2

```

End If

If PLC_I%(41) = 0 Then 'Numbering Web break status
    FrmIstatus!Mh3dLabel17.FillColor = &HFF 'Red
Else
    FrmIstatus!Mh3dLabel17.FillColor = &H80FF00 'Green
End If

If PLC_I%(18) = 0 Then 'Sheeter Web break status
    FrmIstatus!Mh3dLabel18.FillColor = &HFF 'Red
Else
    FrmIstatus!Mh3dLabel18.FillColor = &H80FF00 'Green
End If

If PLC_I%(19) = 0 Then 'Wiping Web break status
    FrmIstatus!Mh3dLabel32.FillColor = &HFF 'Red
Else
    FrmIstatus!Mh3dLabel32.FillColor = &H80FF00 'Green
End If

If PLC_I%(44) = 0 Then 'Sheeter Jamup status
    FrmIstatus!Mh3dLabel19.FillColor = &HFF 'Red
Else
    FrmIstatus!Mh3dLabel19.FillColor = &H80FF00 'Green
End If

If PLC_I%(44) = 0 Then 'Folder Jamup status
    FrmIstatus!Mh3dLabel10.FillColor = &HFF 'Red
Else
    FrmIstatus!Mh3dLabel10.FillColor = &H80FF00 'Green
End If

If PLC_I%(53) <> 0 Then 'Unwinder Core sensor status
    FrmIstatus!Mh3dLabel14.FillColor = &HFF 'Red
Else
    FrmIstatus!Mh3dLabel14.FillColor = &H80FF00 'Green
End If

If PLC_I%(20) <> 0 Then 'Splice detector status
    FrmIstatus!Mh3dLabel15.FillColor = &HFF 'Red
Else
    FrmIstatus!Mh3dLabel15.FillColor = &H80FF00 'Green
End If

If PLC_I%(28) <> 0 Then 'Shuttle motor #1 overtemp status
    FrmIstatus!Mh3dLabel16.FillColor = &HFF 'Red
Else
    FrmIstatus!Mh3dLabel16.FillColor = &H80FF00 'Green
End If

If PLC_I%(29) <> 0 Then 'Shuttle motor #2 overtemp status
    FrmIstatus!Mh3dLabel17.FillColor = &HFF 'Red
Else
    FrmIstatus!Mh3dLabel17.FillColor = &H80FF00 'Green
End If

If PLC_I%(0) <> 0 Then 'System ESTOP status
    FrmIstatus!Mh3dLabel18.FillColor = &HFF 'Red
Else
    FrmIstatus!Mh3dLabel18.FillColor = &H80FF00 'Green
End If

If PLC_O%(0) <> 0 Then 'Unwinder Dancer position status
    FrmIstatus!Mh3dLabel19.FillColor = &HFF 'Red
Else
    FrmIstatus!Mh3dLabel19.FillColor = &H80FF00 'Green
End If

```

FrmIstatus - 3

```

If PLC_I%(54) = 0 Then 'Delivery Piler Down LS status
    FrmIstatus!Mh3dLabel20.FillColor = &HFF 'Red
Else
    FrmIstatus!Mh3dLabel20.FillColor = &H80FF00 'Green
End If

If (0 = 0) Then 'At least one entry in perf table?
    FrmIstatus!Mh3dLabel31.FillColor = &HFF 'Red
Else
    FrmIstatus!Mh3dLabel31.FillColor = &H80FF00 'Green
End If

If PLC_I%(45) = 0 Then 'Intaglio guard status
    FrmIstatus!Mh3dLabel11.FillColor = &HFF 'Red
Else
    FrmIstatus!Mh3dLabel11.FillColor = &H80FF00 'Green
End If

If PLC_I%(46) = 0 Then 'Numbering guard status
    FrmIstatus!Mh3dLabel12.FillColor = &HFF 'Red
Else
    FrmIstatus!Mh3dLabel12.FillColor = &H80FF00 'Green
End If

If PLC_I%(47) = 0 Then 'Sheeter guard status
    FrmIstatus!Mh3dLabel13.FillColor = &HFF 'Red
Else
    FrmIstatus!Mh3dLabel13.FillColor = &H80FF00 'Green
End If

If PLC_O%(0) = 0 Then 'Machine Power status
    FrmIstatus!Mh3dLabel21.FillColor = &HFF 'Red
Else
    FrmIstatus!Mh3dLabel21.FillColor = &H80FF00 'Green
End If

If PLC_O%(39) = 0 Then 'Product Vacuum Blower status
    FrmIstatus!Mh3dLabel22.FillColor = &HFF 'Red
Else
    FrmIstatus!Mh3dLabel22.FillColor = &H80FF00 'Green
End If

If PLC_O%(40) = 0 Then 'Wiping Vacuum Blower status
    FrmIstatus!Mh3dLabel23.FillColor = &H80FFFF 'Yellow
Else
    FrmIstatus!Mh3dLabel23.FillColor = &H80FF00 'Green
End If

If PLC_I%(21) = 0 Then 'Hydraulic Pressure status
    FrmIstatus!Mh3dLabel25.FillColor = &HFF 'Red
Else
    FrmIstatus!Mh3dLabel25.FillColor = &H80FF00 'Green
End If

If PLC_O%(41) = 0 Then 'Scrap Blower status status
    FrmIstatus!Mh3dLabel26.FillColor = &H80FFFF 'Yellow
Else
    FrmIstatus!Mh3dLabel26.FillColor = &H80FF00 'Green
End If

If ProductCalibrated% = 255 Then 'Product Calibrated
    FrmIstatus!Mh3dLabel30.FillColor = &HFF8080 'Blue
Else
    FrmIstatus!Mh3dLabel30.FillColor = &HE0E0E0 'Gray
End If

```

```
FrmIstatus - 4  
If WipingCalibrated% = 255 Then 'Wiping Calibrated  
    FrmIstatus!Mh3dLabel131.FillColor = &HFF8080 'Blue  
    Else  
        FrmIstatus!Mh3dLabel131.FillColor = &HE0E0E0 'Gray  
    End If  
End Sub
```

```

frmDiags = 1
Option Explicit

Private Sub AccessServo1()
    If EnableHostPolling1 = True Then
        If EnableHostPolling2 = False Then
            frmDiags!DMCTerminal2.DMCPollController = False
            frmDiags!DMCTerminal2.Visible = False
            EnableHostPolling2 = True
        End If
        'If the other Servo window was open, close it.
        frmDiags!DMCTerminal1.Visible = True
        frmDiags!DMCTerminal1.DMCPollController = True
        EnableHostPolling1 = False
    Else
        frmDiags!DMCTerminal1.DMCPollController = False
        frmDiags!DMCTerminal1.Visible = False
        EnableHostPolling1 = True
    End If
End Sub

Private Sub AccessServo2()
    If EnableHostPolling2 = True Then
        If EnableHostPolling1 = False Then
            frmDiags!DMCTerminal1.DMCPollController = False
            frmDiags!DMCTerminal1.Visible = False
            EnableHostPolling1 = True
        End If
        'If the other Servo window was open, close it.
        frmDiags!DMCTerminal2.Visible = True
        frmDiags!DMCTerminal2.DMCPollController = True
        EnableHostPolling2 = False
    Else
        frmDiags!DMCTerminal2.DMCPollController = False
        frmDiags!DMCTerminal2.Visible = False
        EnableHostPolling2 = True
    End If
End Sub

Private Sub Command1_Click()
    InitDForce
End Sub

Private Sub Command10_Click()
    ExitDiags
End Sub

Private Sub Command11_Click()
    'Danger: Stops all system timing !!!
    frmMain!Timer2.Enabled = False
    frmMain!Timer3.Enabled = False
    frmMain!Timer4.Enabled = False
    frmDiags!Label24.Visible = True
    frmDiags!Label41.Visible = False
End Sub

Private Sub Command12_Click()

```

Form1.Hide = True

MsgBox: "Shutdown all system timing !!!"

```

Form1.Timer0.Enabled = True
Form1.Timer1.Enabled = True
Form1.Timer2.Enabled = True
Form1.Timer3.Enabled = True
Form1.Timer4.Enabled = True
Form1.Timer5.Enabled = True

```

End Sub

Private Sub Command2\_Click()

AccessServo1

End Sub

Private Sub Form1\_Load()

```

Form1.Visible = False
Timer0.Enabled = False
Timer1.Enabled = False

```

```

If (SystemStartup And 5) = 0 Then
    Form1.Timer0.Enabled = True
End If

```

End Sub

Private Sub Command3\_Click()

AccessServo2

End Sub

Private Sub Command4\_Click()

DisablePLC

End Sub

Private Sub Command5\_Click()

EnablePLC

End Sub

Private Sub Form\_KeyDown(KeyCode As Integer, Shift As Integer)

```

If KeyCode = 4870 Then Call Timer0_Click
If KeyCode = 4871 Then Call AccessServo1
If KeyCode = 4872 Then Call AccessServo2
If KeyCode = 4873 Then Call DisablePLC
If KeyCode = 4874 Then Call EnablePLC
If KeyCode = 4875 Then Call Form1_Load

```

End Sub

Private Sub Form1\_KeyPress()

```

Form1.Visible = True
Form1.Text1.Visible = True
Form1.Text1.Text = ""
Form1.SetFocus

```

End Sub

```

FormHeight = 3

Private Sub Form_Load()
    Dim IV
    For IV = 0 To 2
        FormHeight.MSF(MSC:101).RowHeight(IV) = 218
        FormHeight.MSF(MSC:102).RowHeight(IV) = 218
    Next IV

    For IV = 0 To 7
        FormHeight.MSF(MSC:103).ColWidth(IV) = 428
        FormHeight.MSF(MSC:104).ColWidth(IV) = 428
    Next IV
End Sub

Private Sub Text1_KeyPress(KeyCode As Integer, Shift As Integer)
    Dim L$, R$, M$, M2
    If KeyCode = 8 Then
        If Len(Text1.Text) > 1 Then
            L$ = Instr(1, Text1.Text, "#") 'Look for the delimiter
            If L$ < 0 Then
                R$ = Left(Text1.Text, L$ - 1) 'This is the channel #
                M$ = Mid(Text1.Text, L$ + 1, 2) 'This is the new state
                If Text1(2, "01", M$) Then 'Write zero value receiving number
                    RV = Val(M$) 'Integer type conversion
                Else
                    IV = 0
                    KV = 2 'This will force error
                End If
            Else
                If JV <= 64 And KV <= 1 Then
                    Call PlotOut(RV, KV) 'Write the output
                Else
                    IV = 0
                End If
            End If
        End If
        If L$ = 0 Then
            MsgBox "Invalid Parameter(s)"
            Text1.Text = ""
        End If
    Else
        Label23.Visible = False
        Text1.Visible = False
    End If
End Sub

End Sub

Private Sub Timer1_Timer()
    Call UpdateInputs 'Update the digital inputs
    Call UpdateOutputs 'Update the digital output state
End Sub

Private Sub UpdateInputs()
    Dim L$, R$, M$, M2
    RV = 0 'Initialization output pointer to 0
    KV = 0 'Initialization row pointer to 0
    For IV = 0 To 7 'Loop for all inputs
        RV = PlotIn(IV) 'Read the input
    Next IV

```

```

Image = 0

Image.MaskedOut.Text = "ON"
Image.MaskedOut.Text = "OFF"

If JV = 0 Then
    Image.MaskedOut.Text = "OFF" While the appropriate value
Else
    Image.MaskedOut.Text = "ON"
End If

KJ = KJ + 1 'Increment the column pointer
If KJ > 7 Then
    KJ = 0 'Reset column pointer
    LJ = LJ + 1 'Increment the row pointer
End If
Next JV

End Sub

Private Sub UpdateOutput()
    Dim LK, JK, KJ, LJ, KJ, LJ

    KJ = 0 'Initialize column pointer to 0
    LJ = 0 'Initialize row pointer to 0

    MV = 1 'Start mask at 1
    For LV = 0 To 255 Step 1 'Loop for all outputs
        Image.MaskedOut.Text = "ON" While the appropriate value
        Image.MaskedOut.Text = "OFF"

        If JV = 0 Then
            Image.MaskedOut.Text = "OFF" While the appropriate value
        Else
            Image.MaskedOut.Text = "ON"
        End If
        KJ = KJ + 1 'Increment the column pointer
        MV = MV * 2 'Shift the mask bit left
        If KJ > 7 Then
            KJ = 0 'Reset column pointer
            LJ = LJ + 1 'Increment the row pointer
            MV = 1 'Reset the mask to 1
        End If
    Next LV

End Sub

Private Sub EnablePLC()
    Image.MaskedOut.Text = "ON"
    PLC_RunMode = 1

End Sub

Private Sub DisablePLC()
    Image.MaskedOut.Text = "OFF"
    PLC_RunMode = 0

End Sub

```



```

frmConfig - 1
Option Explicit
Private Sub Command1_Click()
    Call Newfile
End Sub
Private Sub Command10_Click()
    ExitConfig
End Sub

Private Sub Command11_Click()
    'IP adjust negative 0.001
    frmConfig!MhRealInput6.Text = frmConfig!MhRealInput6.Text - 0.001
End Sub

Private Sub Command12_Click()
    'Update the IP calibration
    If frmConfig!Label12.Visible = True Then
        IP_offset# = 0
        frmConfig!MhRealInput6.Text = 0
        VarChangeFlag$(41) = 2
        frmConfig!Label12.Visible = True
    End If
End Sub

Private Sub Command12_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If frmConfig!Label12.Visible = True Then
        frmConfig!Label12.Visible = True
    End If
End Sub

Private Sub Command12_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
    frmConfig!Label12.Visible = False
End Sub

Private Sub Command13_Click()
    VarChangeFlag$(50) = 1
    'Clears the product calibrate flag
End Sub

Private Sub Command2_Click()
    Call Openfile
End Sub

Private Sub Command28_Click()
    'IP adjust posative 0.050

```

```

frmConfig - 2
    frmConfig!MhRealInput6.Text = frmConfig!MhRealInput6.Text + 0.05
End Sub

Private Sub Command29_Click()
    'IP adjust negative 0.050
    frmConfig!MhRealInput6.Text = frmConfig!MhRealInput6.Text - 0.05
End Sub

Private Sub Command3_Click()
    Call SaveFile
End Sub

Private Sub Command30_Click()
    'IP adjust positive 0.001
    frmConfig!MhRealInput6.Text = frmConfig!MhRealInput6.Text + 0.001
End Sub

Private Sub Command31_Click()
    'Initializes the IF calibration
    IP_offset# = 0
    frmConfig!MhRealInput6.Text = 0
    VarChangeFlag$(41) = 1
    frmConfig!Label12.Visible = True
End Sub

Private Sub Command4_Click()
    Call SaveAsFile
End Sub

Private Sub ExitConfig()
    Dim I%
    ConfigTrace% = 0 'Clear return path
    frmConfig.Visible = False
    frmMain!Text1.Text = "Standby"
End Sub

Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
    If KeyCode = &H70 Then Call Newfile
    If KeyCode = &H71 Then Call Openfile
    If KeyCode = &H72 Then Call SaveFile
    If KeyCode = &H73 Then Call SaveAsFile
    If KeyCode = &H79 Then Call ExitConfig
End Sub

Private Sub Newfile()
    'This procedure initializes all variables for a new
    'job.

```

```

frmConfig - 3

Dim I%

WhoseFocus% = 5 'Tell Apad to come back to us
Configtrace% = 1 'Return path

frmApad!Text1.Text = ""
frmApad.Visible = True

End Sub

Private Sub Openfile()

'This procedure loads an existing job profile from disk

Dim I%

WhoseFocus% = 5 'Tell Apad to come back to us
Configtrace% = 2 'Return path

frmApad!Text1.Text = ""
frmApad.Visible = True

End Sub

Private Sub SaveAsFile()

'This procedure saves the current job profile data to a
'user defined filename.

Dim I%

WhoseFocus% = 5 'Tell Apad to come back to us
Configtrace% = 4 'Return path

frmApad!Text1.Text = ""
frmApad.Visible = True

End Sub

Private Sub SaveFile()

'This procedure saves the current job profile to disk

Dim I%
I% = SaveJobdata%()

End Sub

Private Sub MhRealInput1_Click()

WhoseFocus% = 5 'Caller ID
Configtrace% = 9 'Cutoff window end
frmNpad.Visible = True
frmNpad!Text1.Text = "" 'Clear buffer

End Sub

Private Sub MhRealInput10_Click()

WhoseFocus% = 5 'Caller ID
Configtrace% = 5 'Intaglio window end
frmNpad.Visible = True
frmNpad!Text1.Text = "" 'Clear buffer

End Sub

```

```

frmConfig - 4

Private Sub MhRealInput12_Click()
    WhoseFocus% = 5 'Caller ID
    ConfigTrace% = 7 'Discutting window end
    frmNpad.Visible = True
    frmNpad!Text1.Text = "" 'Clear buffer
End Sub

Private Sub MhRealInput15_Click()
    WhoseFocus% = 5 'Caller ID
    ConfigTrace% = 10 'Begin winder torque
    frmNpad.Visible = True
    frmNpad!Text1.Text = "" 'Clear buffer
End Sub

Private Sub MhRealInput16_Click()
    WhoseFocus% = 5 'Caller ID
    ConfigTrace% = 11 'End winder torque
    frmNpad.Visible = True
    frmNpad!Text1.Text = "" 'Clear buffer
End Sub

Private Sub MhRealInput3_Click()
    WhoseFocus% = 5 'Caller ID
    ConfigTrace% = 4 'Intaglio window begin
    frmNpad.Visible = True
    frmNpad!Text1.Text = "" 'Clear buffer
End Sub

Private Sub MhRealInput4_Click()
    WhoseFocus% = 5 'Caller ID
    ConfigTrace% = 6 'Discutting window begin
    frmNpad.Visible = True
    frmNpad!Text1.Text = "" 'Clear buffer
End Sub

Private Sub MhRealInput5_Click()
    WhoseFocus% = 5 'Caller ID
    ConfigTrace% = 8 'Cutoff window begin
    frmNpad.Visible = True
    frmNpad!Text1.Text = "" 'Clear buffer
End Sub

Private Sub MhRealInput6_Change()
    'Handle IP calibration changes

    Dim I#, J#

    J# = frmConfig!MhRealInput6.Text - IP_offset# 'Magnitude of change
    IP_Move# = MasCPI!(7) + J# 'Convert to counts
    ValChangeFlag%(42) = 1 'Make the change
    IP_offset# = frmConfig!MhRealInput6.Text
End Sub

```

frmConfig - 5

Private Sub MhRealInput7\_Click()

WhoseFocus? = 5 'Caller ID  
Configtrace? = 1 'System circumference  
frmNpad.Visible = True  
frmNpad!Text1.Text = "" 'Clear buffer

End Sub

Private Sub MhRealInput8\_Click()

WhoseFocus? = 5 'Caller ID  
Configtrace? = 2 'Numbering circumference  
frmNpad.Visible = True  
frmNpad!Text1.Text = "" 'Clear buffer

End Sub

Private Sub MhRealInput9\_Click()

WhoseFocus? = 5 'Caller ID  
Configtrace? = 3 'Cutoff circumference  
frmNpad.Visible = True  
frmNpad!Text1.Text = "" 'Clear buffer

End Sub

```
frmApad - 1
Option Explicit
Private Sub Command1_Click()
    frmApad!Text1.Text = frmApad!Text1.Text & "6"
End Sub
Private Sub Command10_Click()
    frmApad!Text1.Text = frmApad!Text1.Text & "4"
End Sub
Private Sub Command11_Click()
    frmApad!Text1.Text = frmApad!Text1.Text & "5"
End Sub
Private Sub Command12_Click()
    frmApad!Text1.Text = frmApad!Text1.Text & "1"
End Sub
Private Sub Command13_Click()
    frmApad!Text1.Text = frmApad!Text1.Text & "2"
End Sub
Private Sub Command14_Click()
    frmApad!Text1.Text = frmApad!Text1.Text & "U"
End Sub
Private Sub Command15_Click()
    frmApad!Text1.Text = frmApad!Text1.Text & "Y"
End Sub
Private Sub Command16_Click()
    frmApad!Text1.Text = frmApad!Text1.Text & "T"
End Sub
Private Sub Command17_Click()
    frmApad!Text1.Text = frmApad!Text1.Text & "R"
End Sub
Private Sub Command18_Click()
    frmApad!Text1.Text = frmApad!Text1.Text & "E"
End Sub
Private Sub Command19_Click()
    frmApad!Text1.Text = frmApad!Text1.Text & "W"
End Sub
```

frmApad - 2

Private Sub Command2\_Click()

    frmApad!Text1.Text = frmApad!Text1.Text & "7"

End Sub

Private Sub Command20\_Click()

    frmApad!Text1.Text = frmApad!Text1.Text & "="

End Sub

Private Sub Command21\_Click()

    frmApad!Text1.Text = frmApad!Text1.Text & "+"

End Sub

Private Sub Command22\_Click()

    frmApad!Text1.Text = frmApad!Text1.Text & "P"

End Sub

Private Sub Command23\_Click()

    frmApad!Text1.Text = frmApad!Text1.Text & "O"

End Sub

Private Sub Command24\_Click()

    frmApad!Text1.Text = frmApad!Text1.Text & "I"

End Sub

Private Sub Command26\_Click()

    frmApad!Text1.Text = frmApad!Text1.Text & "A"

End Sub

Private Sub Command27\_Click()

    frmApad!Text1.Text = frmApad!Text1.Text & "S"

End Sub

Private Sub Command28\_Click()

    frmApad!Text1.Text = frmApad!Text1.Text & "D"

End Sub

Private Sub Command29\_Click()

    frmApad!Text1.Text = frmApad!Text1.Text & "F"

End Sub

Private Sub Command3\_Click()

    frmApad!Text1.Text = frmApad!Text1.Text & "8"

End Sub

```
frmApad - 3
Private Sub Command30_Click()
    frmApad!Text1.Text = frmApad!Text1.Text & "G"
End Sub
Private Sub Command31_Click()
    frmApad!Text1.Text = frmApad!Text1.Text & "H"
End Sub
Private Sub Command32_Click()
    frmApad!Text1.Text = frmApad!Text1.Text & "J"
End Sub
Private Sub Command33_Click()
    frmApad!Text1.Text = frmApad!Text1.Text & "K"
End Sub
Private Sub Command34_Click()
    frmApad!Text1.Text = frmApad!Text1.Text & "L"
End Sub
Private Sub Command36_Click()
    Apad_Text$ = frmApad!Text1.Text
    frmApad.Visible = False
    Call ProcessData
End Sub
Private Sub Command4_Click()
    frmApad!Text1.Text = frmApad!Text1.Text & "9"
End Sub
Private Sub Command40_Click()
    frmApad!Text1.Text = frmApad!Text1.Text & ":"
End Sub
Private Sub Command41_Click()
    frmApad!Text1.Text = frmApad!Text1.Text & "."
End Sub
Private Sub Command42_Click()
    frmApad!Text1.Text = frmApad!Text1.Text & ","
End Sub
Private Sub Command43_Click()
    frmApad!Text1.Text = frmApad!Text1.Text & "M"
End Sub
```



frmApad - 4

```
Private Sub Command44_Click()
    frmApad!Text1.Text = frmApad!Text1.Text & "N"
End Sub

Private Sub Command45_Click()
    frmApad!Text1.Text = frmApad!Text1.Text & "B"
End Sub

Private Sub Command46_Click()
    frmApad!Text1.Text = frmApad!Text1.Text & "V"
End Sub

Private Sub Command47_Click()
    frmApad!Text1.Text = frmApad!Text1.Text & "C"
End Sub

Private Sub Command48_Click()
    frmApad!Text1.Text = frmApad!Text1.Text & "X"
End Sub

Private Sub Command49_Click()
    frmApad!Text1.Text = frmApad!Text1.Text & "Z"
End Sub

Private Sub Command5_Click()
    frmApad!Text1.Text = frmApad!Text1.Text & "Q"
End Sub

Private Sub Command55_Click()
    frmApad!Text1.Text = frmApad!Text1.Text & " "
End Sub

Private Sub Command6_Click()
    frmApad!Text1.Text = frmApad!Text1.Text & "Q"
End Sub

Private Sub Command7_Click()
    Dim I%
    I% = Len(frmApad!Text1.Text)
    If I% < 1 Then I% = 1 'Prevent underflow.
    frmApad!Text1.Text = Left$(frmApad!Text1.Text, I% - 1)
End Sub

Private Sub Command9_Click()
```

```

frmApad - 5

    frmApad!Text1.Text = frmApad!Text1.Text & "3"
End Sub

Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
    Dim I%

    If KeyCode = 8 Then
        I% = Len(frmApad.Text1.Text)
        If I% < 1 Then I% = 1 'Don't underrun the buffer
        frmApad.Text1.Text = Left$(frmApad!Text1.Text, I% - 1)
    Else
        If KeyCode = &HD Then 'Did they press Enter key
            Apad_Text$ = frmApad!Text1.Text
            frmApad.Visible = False
            Call ProcessData
        Else
            frmApad!Text1.Text = frmApad!Text1.Text & Chr$(KeyCode)
        End If
    End If
End Sub

End Sub

Private Sub ProcessData()
    'This routine handles the incoming data
    Dim I%

    Select Case WhoseFocus% 'This is the ID of the calling routine
        Case 1 ' * * * * Calls from frmPressrun * * * *
        Case 2 ' * * * * Calls from frmP00Flexo * * * *
            Select Case P00Flexotrace%
                Case 3 'Returned with filename for Job Open
                    If Apad_Text$ <> "" Then 'Did we get a value
                        frmMain.Text6.Text = Apad_Text$
                        I% = LoadJobdata%()
                    End If
                Case 4 'Incoming filename for SaveAs
                    If Apad_Text$ <> "" Then 'Value???
                        frmMain.Text6.Text = Apad_Text$
                        I% = SaveJobdata%()
                    End If
                Case 5 'Filename for Newfile function
                    If Apad_Text$ <> "" Then 'Value???
                        frmP00Flexo.Text1.Text = Apad_Text$
                        I% = NewJobData%()
                    End If
            End Select
        End Select 'End of Case 2

        Case 3 ' * * * * Parameters calls * * * *
            frmPtable.MSFlexGrid1.Text = Apad_Text$
    End Select

```

frmApad - 6

```
Case 4 ' * * * * Production calls * * * *

Select Case Productiontrace%

Case 2 'This is the operator ID
  If Apad_Text$ <> "" Then
    FrmProduction.Text1.Text = Apad_Text$
    frmMain.Text2.Text = Apad_Text$
  End If

End Select 'End of Case 4

Case 5 ' * * * * Calls from frmConfig screen * * * *
Select Case Configtrace%

Case 1 'Filename for Newfile function
  If Apad_Text$ <> "" Then 'Value???
    frmMain.Text6.Text = Apad_Text$
    I% = NewJobData%()
  End If

Case 2 'Returned with filename for Job Open
  If Apad_Text$ <> "" Then 'Did we get a value
    frmMain.Text6.Text = Apad_Text$
    I% = LoadJobdata%()
  End If

Case 4 'Incoming filename for SaveAs
  If Apad_Text$ <> "" Then 'Value???
    frmMain.Text6.Text = Apad_Text$
    I% = SaveJobdata%()
  End If

End Select 'End of Case 5

Case 6 ' * * * * Calls from frmP00Intaglio

End Select

End Sub
```

## Module5 - 1

```
'Program      : General
'Creation date : 20-Oct-00
'Module       : AIODRV.BAS
'Version      : V1.0
'Edit        : -00
'Edit date   : 25-Oct-00
'Author      : Joseph B. Schutte III
              : Industrial Light & Motion, Inc.
              : 2170 Van Blaricum Rd.
              : Cincinnati OH 45233 USA
```

```
'Copyright (C) 2000 Industrial Light & Motion, Inc.
```

```
'Revision History
```

```
'Ver/Edit   Edit date   Who      Reason
```

```
'This module handles the I/O mapping through the AIO
'card. Two primary operations are supported:
```

```
'AioInit - Sub that initializes the AIO board and
           working variables.
'AioIn$(x) - Function that returns the value of an
            input defined by x. (NOT SUPPORTED IN THIS CARD)
'AioOut(x,y) - Sub that sets the Output defined by X to
              the value y (bipolar).
```

```
'Analog Output assignments
```

Channel	Description
0	Main Drive velocity command
1	Wiping Drive velocity command
2	Wiping Rewinder velocity command
3	Intaglio Inker #1 velocity command
4	Intaglio Inker #2 velocity command
5	Product Rewinder velocity command
6	Not assigned
7	Not assigned

```
Global AnOut$(128) 'Analog OUT image registers
Global AnIn$(128) 'Analog IN image registers
```

```
Const IobaseAddr% = &HEF00 'This is the hardware base address
Const IoCalAddr% = &HE800 'This is the address of calibration reg
```

```
Declare Function MhInpByte Lib "MhMu32.DLL"
                        (ByVal Port As Long) As Byte
```

```
Declare Function MhInpWord Lib "MhMu32.DLL"
                        (ByVal Port As Long) As Long
```

```
Declare Sub MhOutByte Lib "MhMu32.DLL"
                        (ByVal Port As Long, ByVal Valu As Byte)
```

```
Declare Sub MhOutWord Lib "MhMu32.DLL"
                        (ByVal Port As Long,
                        ByVal Valu As Integer)
```

```
Function AioIn$(Addr%)
```

```
Dim I%
```

```
'Read the Analog Input register
```

Module5 - 2

```
'This function not supported in this version software
```

```
'I% = MhInpByte(Addr%) 'Read the Hardware input
```

```
End Function
```

```
Public Sub AioInit()
```

```
'This routine initializes the Analog output board. additional code  
'will be placed here to support Analog inputs.
```

```
Dim I%
```

```
I% = MhInpByte(IobaseAddr% + 10)
```

```
'This read sets the card to Automatic update mode
```

```
I% = MhInpByte(IobaseAddr% + 15)
```

```
'This read unrestricts the output voltage range
```

```
For I% = 0 To 8
```

```
    Call AioOut(I%, 0)
```

```
Next I%
```

```
End Sub
```

```
Public Sub AioOut(Channel%, NuVal%)
```

```
Dim H%, L%
```

```
AnOut%(Channel%) = NuVal% 'Save the original value
```

```
If NuVal% > 2047 Then NuVal% = 2047
```

```
If NuVal% < -2047 Then NuVal% = -2047
```

```
Call MhOutWord(IobaseAddr% + (Channel% * 2), NuVal% + &H800)
```

```
End Sub
```

## Module4 - 1

```
'Program      : General library module
'Creation date : 19-Oct-93
'Module       : SERVODRV.BAS
'Version      : V3.0
'Edit        : -01
'Edit date   : 26-Sep-00
'Author      : Joseph B. Schutte III
              : Industrial Light & Motion, Inc.
              : 2170 Van Blaricum Rd.
              : Cincinnati OH 45233 USA
'
' Copyright (C) 1993-2000 Industrial Light & Motion, Inc.
```

## 'Revision History

Ver/Ed	Edit date	Who	Reason
'V3.1-01	22-Sep-00	JBSIII	Mod code to access DMC-1080's (dual)
'V3.0-00	02-Jan-00	JBSIII	Revamp code to support 2100 series and dual motion processors, and various bugs
'V2.1-02	27-Oct-99	JBSIII	Minor mods to support Ethernet based 2100 series controllers
'V2.1-01	10-Jul-97	JBSIII	Add Boot Galil boot loader
'V2.0-00	12-Feb-97	JBSIII	Major upgrade 32 bit / Win95 & NT
'V1.0-01	28-Aug-94	JBSIII	Sense Estop & Reinit

## Option Explicit

## Function CmdServo1%(S\_command\$)

```
Dim I%, A$

frmMain.DMCShell1.DMCCCommand = S_command$
A$ = LeftB$(frmMain.DMCShell1.DMCResponse, 1)
If A$ = "?" Then
    CmdServo1% = -1 'Error occurred
Else
    CmdServo1% = 0 'Transaction OK
End If
```

End Function

## Function CmdServo2%(S\_command\$)

```
Dim I%, A$

frmMain.DMCShell2.DMCCCommand = S_command$
A$ = LeftB$(frmMain.DMCShell2.DMCResponse, 1)
If A$ = "?" Then
    CmdServo2% = -1 'Error occurred
Else
    CmdServo2% = 0 'Transaction OK
End If
```

End Function

## Function GetServoErcode1%(SelAxis%)

```
' Get the Motion status of the specified Axis ( Processor A )

Dim I%, J%

I% = CmdServo1%("TC") 'Servo status command

If I% <> 0 Then 'We received an error
    I% = -1 'Error flag
```

Module4 - 2

```

Else
End If

GetServoErcode1% = I%

End Function

Function GetServoErcode2%(SelAxis%)
' Get the Motion status of the specified Axis ( Processor B )
Dim I%, J%
I% = CmdServo2%("TC") 'Servo status command
If I% <> 0 Then 'We received an error
    I% = -1 'Error flag
Else
End If

GetServoErcode2% = I%

End Function

Function GetServoStatus%(SelAxis%)
' Get the Motion status of the specified Axis
Dim I%, J%
Select Case SelAxis% 'Vector to the requested Axis
    Case 0: I% = CmdServo1%("SC X") 'Axis 0
    Case 1: I% = CmdServo1%("SC Y") 'Axis 1
    Case 2: I% = CmdServo1%("SC Z") 'Axis 2
    Case 3: I% = CmdServo1%("SC W") 'Axis 3
    Case 4: I% = CmdServo1%("SC E") 'Axis 4
    Case 5: I% = CmdServo1%("SC F") 'Axis 5
    Case 6: I% = CmdServo1%("SC G") 'Axis 6
    Case 7: I% = CmdServo1%("SC H") 'Axis 7
    Case 8: I% = CmdServo2%("SC X") 'Axis 8
    Case 9: I% = CmdServo2%("SC Y") 'Axis 9
    Case 10: I% = CmdServo2%("SC Z") 'Axis 10
    Case 11: I% = CmdServo2%("SC W") 'Axis 11
    Case 12: I% = CmdServo2%("SC E") 'Axis 12
    Case 13: I% = CmdServo2%("SC F") 'Axis 13
    Case 14: I% = CmdServo2%("SC G") 'Axis 14
    Case 15: I% = CmdServo2%("SC H") 'Axis 15
    Case Else: I% = -1
End Select

If I% <> 0 Then 'We received an error
    I% = -1 'Error flag
Else
    If I% <= 7 Then
        J% = InStr(1, frmMain.DMCShell1.DMCResponse, Chr$(13)) 'Find the delimiter
        I% = CInt(LeftB$(frmMain.DMCShell1.DMCResponse, J% - 1)) 'This is the returned statu
s from servo
    Else
        J% = InStr(1, frmMain.DMCShell2.DMCResponse, Chr$(13)) 'Find the delimiter

```

Module4 - 3

```

    I% = CInt(LeftB$(frmMain.DMCShell2.DMCResponse, J% - 1)) 'This is the returned statu
s from servo
    End If
    End If

    GetServoStatus% = I%

End Function

Sub InitPhase()

```

```

    'Set the Encoder and Motor phase here per Parameters

```

```

    Dim I%, J%, K%, L$, M$, N$, O$, Enc%, Mtr%

```

```

    J% = CInt(Param tbl$(10, 2)) - 1 'Number of active axis'

```

```

    If K% <> 0 Then 'We have a command error
        MotionSystemStatus% = 255 'Fault flag
        MsgBox ("?Cmd Error during INIT-Phase " & Str$(I%))
    End If

```

```

    Next I%

```

```

End Sub

```

```

Function InitServo%()

```

```

    Dim base_address%, I%, J%, K%

```

```

    I% = 0

```



Module 4 - 4

Testbypass:

End Function

Sub LoadGalilTuning()

' Initialize the Servo Gain Parameters

Dim I%, J%, K%, L%

L% = CInt(Param tbl\$(10, 2))

For I% = 1 To (L% - 1) 'All axes

If I% &lt;= 7 Then

J% = (I% + 1) \* 100 'Base address for parameters

J% = (I% + 3) \* 100 'Base address with skip to upper addr

End If

K% = WriteServoCmd\$(I%, "GN", Param tbl\$(J% + 16, 2)) 'Load GN

K% = WriteServoCmd\$(I%, "IT", Param tbl\$(J% + 17, 2)) 'Load IT

K% = WriteServoCmd\$(I%, "VT", Param tbl\$(J% + 18, 2)) 'Load VT

K% = WriteServoCmd\$(I%, "FA", Param tbl\$(J% + 19, 2)) 'Load FA

K% = WriteServoCmd\$(I%, "AC", Param tbl\$(J% + 20, 2)) 'Load Accel

K% = WriteServoCmd\$(I%, "DC", Param tbl\$(J% + 20, 2)) 'Load Decel

K% = WriteServoCmd\$(I%, "FV", Param tbl\$(J% + 27, 2)) 'Load FA

K% = WriteServoCmd\$(I%, "KP", Param tbl\$(J% + 13, 2)) 'Load KP

K% = WriteServoCmd\$(I%, "KI", Param tbl\$(J% + 14, 2)) 'Load KI

K% = WriteServoCmd\$(I%, "KD", Param tbl\$(J% + 15, 2)) 'Load KD

K% = WriteServoCmd\$(I%, "PL", Param tbl\$(J% + 33, 2)) 'Load PL

K% = WriteServoCmd\$(I%, "IL", Param tbl\$(J% + 34, 2)) 'Load IL

Next I%

End Sub

Function ReadServoData\$(SelAxis%, OpCode%)

Module4 - 5

```

'Routine issues command to Servo and returns string response
' Get the Motion status of the specified Axis

Dim I%, J%, O$

O$ = Mid$("XYZWEFGHXYZWEFGH", SelAxis% + 1, 1) 'Parse out the axis.ID
If SelAxis% <= 7 Then
    I% = CmdServo1$(OpCode$ & O$) 'Axis 0-7
Else
    I% = CmdServo2$(OpCode$ & O$) 'Axis 8-15
End If

If I% <> 0 Then 'We received an error
    ReadServoData$ = "" 'Error flag
Else
    If SelAxis% <= 7 Then
        J% = InStr(1, frmMain.DMCShell1.DMCResponse, Chr$(13)) 'Find the delimiter
    Else
        J% = InStr(1, frmMain.DMCShell2.DMCResponse, Chr$(13)) 'Find the delimiter
    End If
    If J% <> 0 Then 'We received a message
        If SelAxis% <= 7 Then
            ReadServoData$ = LeftB$(frmMain.DMCShell1.DMCResponse, J% - 1) 'This is the returned status from servo #1
        Else
            ReadServoData$ = LeftB$(frmMain.DMCShell2.DMCResponse, J% - 1) 'This is the returned status from servo #2
        End If
    Else
        ReadServoData$ = "" 'Nothing to reply
    End If
End If

End Function

Function SysPowerCheck%()

'Check Servo Power Supply Status

Dim I%, J%, K%, L%, M$

If (PLC I$(0) <> 0) Then
    SysPowerCheck% = 0
    SystemStatus% = SystemStatus% Or &H100 'Power ON
    frmMain!Label1.Visible = False
Else
    SysPowerCheck% = 1
    SystemStatus% = SystemStatus% And &HFEFF 'Power OFF
    frmMain!Label1.Visible = True
End If

End Function

Function WriteServoCmd$(SelAxis%, OpCode$, Operand$)

' Routine to send a general command to an Axis

Select Case SelAxis% 'Vector to the requested Axis

    Case 0: WriteServoCmd$ = CmdServo1$(OpCode$ & "X=" & Operand$) 'Axis 0
    Case 1: WriteServoCmd$ = CmdServo1$(OpCode$ & "Y=" & Operand$) 'Axis 1
    Case 2: WriteServoCmd$ = CmdServo1$(OpCode$ & "Z=" & Operand$) 'Axis 2
    Case 3: WriteServoCmd$ = CmdServo1$(OpCode$ & "W=" & Operand$) 'Axis 3
    Case 4: WriteServoCmd$ = CmdServo1$(OpCode$ & "E=" & Operand$) 'Axis 4
    Case 5: WriteServoCmd$ = CmdServo1$(OpCode$ & "F=" & Operand$) 'Axis 5

```

Module4 - 6

```

Case 6: WriteServoCmd% = CmdServo1%(OpCode$ & "G=" & Operand$) 'Axis 6
Case 7: WriteServoCmd% = CmdServo1%(OpCode$ & "H=" & Operand$) 'Axis 7
Case 8: WriteServoCmd% = CmdServo2%(OpCode$ & "X=" & Operand$) 'Axis 8
Case 9: WriteServoCmd% = CmdServo2%(OpCode$ & "Y=" & Operand$) 'Axis 9
Case 10: WriteServoCmd% = CmdServo2%(OpCode$ & "Z=" & Operand$) 'Axis 10
Case 11: WriteServoCmd% = CmdServo2%(OpCode$ & "W=" & Operand$) 'Axis 11
Case 12: WriteServoCmd% = CmdServo2%(OpCode$ & "E=" & Operand$) 'Axis 12
Case 13: WriteServoCmd% = CmdServo2%(OpCode$ & "F=" & Operand$) 'Axis 13
Case 14: WriteServoCmd% = CmdServo2%(OpCode$ & "G=" & Operand$) 'Axis 14
Case 15: WriteServoCmd% = CmdServo2%(OpCode$ & "H=" & Operand$) 'Axis 15
Case Else: WriteServoCmd% = -1

```

End Select

End Function

Function WriteServoXcmd%(SelAxis%, OpCode\$)

' Routine to send a command to an Axis without data

Select Case SelAxis% 'Vector to the requested Axis

```

Case 0: WriteServoXcmd% = CmdServo1%(OpCode$ & "X") 'Axis 0
Case 1: WriteServoXcmd% = CmdServo1%(OpCode$ & "Y") 'Axis 1
Case 2: WriteServoXcmd% = CmdServo1%(OpCode$ & "Z") 'Axis 2
Case 3: WriteServoXcmd% = CmdServo1%(OpCode$ & "W") 'Axis 3
Case 4: WriteServoXcmd% = CmdServo1%(OpCode$ & "E") 'Axis 4
Case 5: WriteServoXcmd% = CmdServo1%(OpCode$ & "F") 'Axis 5
Case 6: WriteServoXcmd% = CmdServo1%(OpCode$ & "G") 'Axis 6
Case 7: WriteServoXcmd% = CmdServo1%(OpCode$ & "H") 'Axis 7
Case 8: WriteServoXcmd% = CmdServo2%(OpCode$ & "X") 'Axis 8
Case 9: WriteServoXcmd% = CmdServo2%(OpCode$ & "Y") 'Axis 9
Case 10: WriteServoXcmd% = CmdServo2%(OpCode$ & "Z") 'Axis 10
Case 11: WriteServoXcmd% = CmdServo2%(OpCode$ & "W") 'Axis 11
Case 12: WriteServoXcmd% = CmdServo2%(OpCode$ & "E") 'Axis 12
Case 13: WriteServoXcmd% = CmdServo2%(OpCode$ & "F") 'Axis 13
Case 14: WriteServoXcmd% = CmdServo2%(OpCode$ & "G") 'Axis 14
Case 15: WriteServoXcmd% = CmdServo2%(OpCode$ & "H") 'Axis 15
Case Else: WriteServoXcmd% = -1

```

End Select

End Function

Module4 - 7

```

End Sub
Public Sub Load_app_variables()

    'Routine downloads the application variables held in the parameter
    'file

    Dim I%, J%, K%

    K% = LastAxis%
    If LastAxis% > 7 Then 'Setup for split table read
        K% = 7
    Else
        K% = LastAxis%
    End If

    For I% = 0 To K% 'Need to capture CPI's
        MasCPR!(I%) = CDbl(Param_tbl$((I% + 1) * 100) + 2, 2))
        MasCPI!(I%) = CDbl(Param_tbl$((I% + 1) * 100) + 2, 2)) / CDbl(Param_tbl$((I% + 1) * 10
0) + 32, 2))
    Next I%

    If LastAxis% > 7 Then 'Get the CPI's for the high # axis'

        For I% = 8 To LastAxis% 'Need to capture CPI's
            MasCPR!(I%) = CDbl(Param_tbl$(300 + (I% * 100) + 2, 2))
            MasCPI!(I%) = CDbl(Param_tbl$(300 + (I% * 100) + 2, 2)) / CDbl(Param_tbl$(300 + (I%
* 100) + 32, 2))
        Next I%
    End If

    'MasCPI!(7) = MasCPR!(7) / CDbl(frmSCsetup!MhRealInput4.Text)
    'CPI based on repeat length

    J% = 0
    J% = J% + CmdServo1$("AxisAAC=" & Param_tbl$(120, 2))
    J% = J% + CmdServo1$("AxisBAC=" & Param_tbl$(220, 2))
    J% = J% + CmdServo1$("AxisCAC=" & Param_tbl$(320, 2))
    J% = J% + CmdServo1$("AxisDAC=" & Param_tbl$(420, 2))
    J% = J% + CmdServo1$("AxisEAC=" & Param_tbl$(520, 2))
    J% = J% + CmdServo1$("AxisFAC=" & Param_tbl$(620, 2))
    J% = J% + CmdServo1$("AxisGAC=" & Param_tbl$(720, 2))
    J% = J% + CmdServo1$("AxisHAC=" & Param_tbl$(820, 2))
    'J% = J% + CmdServo2$("AxisIAC=" & Param_tbl$(1120, 2))
    'J% = J% + CmdServo2$("AxisJAC=" & Param_tbl$(1220, 2))
    'J% = J% + CmdServo2$("AxisKAC=" & Param_tbl$(1320, 2))
    'J% = J% + CmdServo2$("AxisLAC=" & Param_tbl$(1420, 2))
    'J% = J% + CmdServo2$("AxisMAC=" & Param_tbl$(1520, 2))
    'J% = J% + CmdServo2$("AxisNAC=" & Param_tbl$(1620, 2))
    'J% = J% + CmdServo2$("AxisOAC=" & Param_tbl$(1720, 2))
    'J% = J% + CmdServo2$("AxisPAC=" & Param_tbl$(1820, 2))

    J% = J% + CmdServo1$("CPRA=" & Param_tbl$(102, 2)) 'Counts per rev
    J% = J% + CmdServo1$("CPRE=" & Param_tbl$(202, 2))
    J% = J% + CmdServo1$("CPRC=" & Param_tbl$(302, 2))
    J% = J% + CmdServo1$("CPRD=" & Param_tbl$(402, 2))

```

Module4 - 8

```

J% = J% + CmdServo1%("CPRE=" & Param_tbl$(502, 2))
J% = J% + CmdServo1%("CPRF=" & Param_tbl$(602, 2))
J% = J% + CmdServo1%("CPRG=" & Param_tbl$(702, 2))
J% = J% + CmdServo1%("CPRH=" & Param_tbl$(802, 2))
J% = J% + CmdServo2%("CPRDX=" & Param_tbl$(802, 2))
J% = J% + CmdServo2%("CPRI=" & Param_tbl$(1102, 2))
J% = J% + CmdServo2%("CPRJ=" & Param_tbl$(1202, 2))
J% = J% + CmdServo2%("CPRK=" & Param_tbl$(1302, 2))
J% = J% + CmdServo2%("CPRL=" & Param_tbl$(1402, 2))
J% = J% + CmdServo2%("CPRM=" & Param_tbl$(1502, 2))
J% = J% + CmdServo2%("CPRN=" & Param_tbl$(1602, 2))
J% = J% + CmdServo2%("CPRO=" & Param_tbl$(1702, 2))
J% = J% + CmdServo2%("CPRP=" & Param_tbl$(1802, 2))

J% = J% + CmdServo1%("AHoffA=" & Param_tbl$(106, 2)) 'ABS Home offsets
J% = J% + CmdServo1%("AHoffB=" & Param_tbl$(206, 2))
J% = J% + CmdServo1%("AHoffC=" & Param_tbl$(306, 2))
J% = J% + CmdServo1%("AHoffD=" & Param_tbl$(406, 2))
J% = J% + CmdServo1%("AHoffE=" & Param_tbl$(506, 2))
J% = J% + CmdServo1%("AHoffF=" & Param_tbl$(606, 2))
J% = J% + CmdServo1%("AHoffG=" & Param_tbl$(706, 2))
J% = J% + CmdServo1%("AHoffH=" & Param_tbl$(806, 2))
J% = J% + CmdServo2%("AHoffI=" & Param_tbl$(1106, 2))
J% = J% + CmdServo2%("AHoffJ=" & Param_tbl$(1206, 2))
J% = J% + CmdServo2%("AHoffK=" & Param_tbl$(1306, 2))
J% = J% + CmdServo2%("AHoffL=" & Param_tbl$(1406, 2))
J% = J% + CmdServo2%("AHoffM=" & Param_tbl$(1506, 2))
J% = J% + CmdServo2%("AHoffN=" & Param_tbl$(1606, 2))
J% = J% + CmdServo2%("AHoffO=" & Param_tbl$(1706, 2))
J% = J% + CmdServo2%("AHoffP=" & Param_tbl$(1806, 2))

J% = J% + CmdServo1%("HoSpA=" & Param_tbl$(111, 2)) 'Homing speeds
J% = J% + CmdServo1%("HoSpB=" & Param_tbl$(211, 2))
J% = J% + CmdServo1%("HoSpC=" & Param_tbl$(311, 2))
J% = J% + CmdServo1%("HoSpD=" & Param_tbl$(411, 2))
J% = J% + CmdServo1%("HoSpE=" & Param_tbl$(511, 2))
J% = J% + CmdServo1%("HoSpF=" & Param_tbl$(611, 2))
J% = J% + CmdServo1%("HoSpG=" & Param_tbl$(711, 2))
J% = J% + CmdServo1%("HoSpH=" & Param_tbl$(811, 2))
J% = J% + CmdServo2%("HoSpI=" & Param_tbl$(1111, 2))
J% = J% + CmdServo2%("HoSpJ=" & Param_tbl$(1211, 2))
J% = J% + CmdServo2%("HoSpK=" & Param_tbl$(1311, 2))
J% = J% + CmdServo2%("HoSpL=" & Param_tbl$(1411, 2))
J% = J% + CmdServo2%("HoSpM=" & Param_tbl$(1511, 2))
J% = J% + CmdServo2%("HoSpN=" & Param_tbl$(1611, 2))
J% = J% + CmdServo2%("HoSpO=" & Param_tbl$(1711, 2))
J% = J% + CmdServo2%("HoSpP=" & Param_tbl$(1811, 2))

J% = J% + CmdServo1%("HoSpzA=" & Param_tbl$(112, 2)) 'Z Ref speeds
J% = J% + CmdServo1%("HoSpzB=" & Param_tbl$(212, 2))
J% = J% + CmdServo1%("HoSpzC=" & Param_tbl$(312, 2))
J% = J% + CmdServo1%("HoSpzD=" & Param_tbl$(412, 2))
J% = J% + CmdServo1%("HoSpzE=" & Param_tbl$(512, 2))
J% = J% + CmdServo1%("HoSpzF=" & Param_tbl$(612, 2))
J% = J% + CmdServo1%("HoSpzG=" & Param_tbl$(712, 2))
J% = J% + CmdServo1%("HoSpzH=" & Param_tbl$(812, 2))
J% = J% + CmdServo2%("HoSpzI=" & Param_tbl$(1112, 2))
J% = J% + CmdServo2%("HoSpzJ=" & Param_tbl$(1212, 2))
J% = J% + CmdServo2%("HoSpzK=" & Param_tbl$(1312, 2))
J% = J% + CmdServo2%("HoSpzL=" & Param_tbl$(1412, 2))
J% = J% + CmdServo2%("HoSpzM=" & Param_tbl$(1512, 2))
J% = J% + CmdServo2%("HoSpzN=" & Param_tbl$(1612, 2))
J% = J% + CmdServo2%("HoSpzO=" & Param_tbl$(1712, 2))
J% = J% + CmdServo2%("HoSpzP=" & Param_tbl$(1812, 2))

J% = J% + CmdServo1%("CtWghtA=" & Str$(MasCPI!(0))) 'Counts per inch
J% = J% + CmdServo1%("CtWghtB=" & Str$(MasCPI!(1)))

```

Module4 - 9

```

J% = J% + CmdServo1%("CtWghtC=" & Str$(MasCPI!(2)))
J% = J% + CmdServo1%("CtWghtD=" & Str$(MasCPI!(3)))
J% = J% + CmdServo1%("CtWghtE=" & Str$(MasCPI!(4)))
J% = J% + CmdServo1%("CtWghtF=" & Str$(MasCPI!(5)))
J% = J% + CmdServo1%("CtWghtG=" & Str$(MasCPI!(6)))
J% = J% + CmdServo1%("CtWghtH=" & Str$(MasCPI!(7)))
J% = J% + CmdServo2%("CtWghtI=" & Str$(MasCPI!(8)))
J% = J% + CmdServo2%("CtWghtJ=" & Str$(MasCPI!(9)))
J% = J% + CmdServo2%("CtWghtK=" & Str$(MasCPI!(10)))
J% = J% + CmdServo2%("CtWghtL=" & Str$(MasCPI!(11)))
'J% = J% + CmdServo2%("CtWghtM=" & Str$(MasCPI!(12)))
'J% = J% + CmdServo2%("CtWghtN=" & Str$(MasCPI!(13)))
'J% = J% + CmdServo2%("CtWghtO=" & Str$(MasCPI!(14)))
'J% = J% + CmdServo2%("CtWghtP=" & Str$(MasCPI!(15)))

J% = J% + CmdServo1%("JogSpeed=" & Param_tbl$(904, 2))
J% = J% + CmdServo2%("JogSpeed=" & Param_tbl$(904, 2))
J% = J% + CmdServo1%("ProdHoS=" & Param_tbl$(908, 2))
J% = J% + CmdServo2%("ProdHoS=" & Param_tbl$(908, 2))
J% = J% + CmdServo1%("WautoonS=" & Param_tbl$(911, 2))
J% = J% + CmdServo1%("WipVBmax=" & Param_tbl$(922, 2))
J% = J% + CmdServo1%("WipeJogS=" & Param_tbl$(925, 2))
J% = J% + CmdServo1%("IPwipcal=" & Param_tbl$(927, 2))
J% = J% + CmdServo1%("WipeHoS=" & Param_tbl$(929, 2))
J% = J% + CmdServo1%("IPwipaut=" & Param_tbl$(930, 2))
J% = J% + CmdServo1%("WVBegain=" & Param_tbl$(931, 2))
J% = J% + CmdServo1%("WVBcalof=" & Param_tbl$(932, 2))
J% = J% + CmdServo1%("VB3nullP=" & Param_tbl$(933, 2))
J% = J% + CmdServo1%("VB4nullP=" & Param_tbl$(934, 2))
J% = J% + CmdServo1%("VB1nullP=" & Param_tbl$(935, 2))
J% = J% + CmdServo1%("VB2nullP=" & Param_tbl$(936, 2))
J% = J% + CmdServo1%("IntMEmax=" & Param_tbl$(937, 2))
J% = J% + CmdServo1%("IntMEgn=" & Param_tbl$(938, 2))
J% = J% + CmdServo1%("IntVBmax=" & Param_tbl$(939, 2))
J% = J% + CmdServo1%("IntVBgn=" & Param_tbl$(940, 2))
J% = J% + CmdServo1%("DieMEmax=" & Param_tbl$(941, 2))
J% = J% + CmdServo1%("DieMEgn=" & Param_tbl$(942, 2))
J% = J% + CmdServo1%("DieVBmax=" & Param_tbl$(943, 2))
J% = J% + CmdServo1%("DieVBgn=" & Param_tbl$(944, 2))
J% = J% + CmdServo1%("DieSave=" & Param_tbl$(945, 2))
J% = J% + CmdServo1%("DieSintv=" & Param_tbl$(946, 2))
J% = J% + CmdServo1%("arVBmax=" & Param_tbl$(947, 2))
J% = J% + CmdServo1%("arVBgn=" & Param_tbl$(948, 2))
J% = J% + CmdServo1%("VB1nullA=" & Param_tbl$(949, 2))
J% = J% + CmdServo2%("Maxerr=" & Param_tbl$(950, 2))
J% = J% + CmdServo1%("IntLwin=" & Param_tbl$(951, 2))
frmConfig!MhRealInput3.Text = Param_tbl$(951, 2)
J% = J% + CmdServo1%("IntUwin=" & Param_tbl$(952, 2))
frmConfig!MhRealInput10.Text = Param_tbl$(952, 2)
J% = J% + CmdServo1%("DieLwin=" & Param_tbl$(953, 2))
frmConfig!MhRealInput4.Text = Param_tbl$(953, 2)
J% = J% + CmdServo1%("DieUwin=" & Param_tbl$(954, 2))
frmConfig!MhRealInput12.Text = Param_tbl$(954, 2)
J% = J% + CmdServo2%("CutLwin=" & Param_tbl$(955, 2))
frmConfig!MhRealInput5.Text = Param_tbl$(955, 2)
J% = J% + CmdServo2%("CutUwin=" & Param_tbl$(956, 2))
frmConfig!MhRealInput1.Text = Param_tbl$(956, 2)
J% = J% + CmdServo2%("CutMEmax=" & Param_tbl$(957, 2))
J% = J% + CmdServo2%("CutMEgn=" & Param_tbl$(958, 2))
J% = J% + CmdServo2%("AxisX[7]=" & Param_tbl$(959, 2))
J% = J% + CmdServo2%("AxisY[7]=" & Param_tbl$(960, 2))
J% = J% + CmdServo2%("AxisZ[7]=" & Param_tbl$(961, 2))

```

JogSpeed% = Param\_tbl\$(904, 2) 'Map for global use'

If J% <> 0 Then 'Error loading variables

Module4 - 10

```
        MsgBox ("?Error loading App variables")
    Else
    End If
```

End Sub

Public Function ReqServo\$(Command\$, Processor%)

```
'New function as of 12-Jul-97 to read data from the motion card.
'Legal commands are passed through Command$ with the response
'returned to the calling function. Errors return null string
```

Dim I%

```
ReqServo$ = ""
Select Case Processor%
```

```
Case 0 ' First processor
I% = CmdServo1$(Command$) 'Request the data
I% = Instr(1, frmMain!DMCShell1.DMCResponse, Chr$(10))
If I% = 0 Then I% = 1 'Return null if LF not found
ReqServo$ = Mid$(frmMain!DMCShell1.DMCResponse, 1, I% - 1)
```

```
Case 1 ' Second processor
I% = CmdServo2$(Command$) 'Request the data
I% = Instr(1, frmMain!DMCShell2.DMCResponse, Chr$(10))
If I% = 0 Then I% = 1 'Return null if LF not found
ReqServo$ = Mid$(frmMain!DMCShell2.DMCResponse, 1, I% - 1)
```

End Select

End Function

## Module3 - 1

```

'Program      : General
'Creation date : 21-Oct-94
'Module       : Piodrv.BAS
'Version      : V2.0
'Edit        : -00
'Edit date    : 12-Feb-97
'Author       : Joseph B. Schutte III
               : Industrial Light & Motion, Inc.
               : 2170 Van Blaricum Rd.
               : Cincinnati OH 45233 USA
'Copyright (C) 1995-2000 Industrial Light & Motion, Inc.
'Revision History
'
'Ver/Edit   Edit date   Who      Reason
'V2.0-01    10-Oct-00   JBSIII   Mod addresses for PCI card
'V2.0-00    12-Feb-97   JBSIII   Major upgrade 32 bit / Win95 & NT
'V1.5-00    19-Jun-95   JBSIII   Add ScanInputs and ScanOutputs subroutine
'
'This module handles the I/O mapping through the PIO
'card. Four primary operations are supported:
'
'PioInit - Sub that initializes the PIO board and
'          working variables.
'PioIn$(x) - Function that returns the condition of an
'           input defined by x.
'PioOut(x,y) - Sub that sets the Input defined by X to
'             the value y (0 or 1).
'
'ScanPLCInputs - Sub reads all inputs and stores in array PLC_Imap$()
'StrobePLCOutputs - Sub writes contents of PLC_Omap$() to the
'                  discrete outputs.
'
Global OutImage$(10) 'Image of Output registers here
Global PLC_I$(128)   'PLC discrete input registers
Global PLC_I_Last$(128) 'Track the values on the last scan
Global PLC_O$(128)   'PLC discrete output registers
'
Const IobaseAddr% = &HEF80 'This is the hardware base address
'
Declare Function MhInpByte Lib "MhMu32.DLL"
'          (ByVal Port As Long) As Byte
'
Declare Function MhInpWord Lib "MhMu32.DLL"
'          (ByVal Port As Long) As Long
'
Declare Sub MhOutByte Lib "MhMu32.DLL"
'          (ByVal Port As Long, ByVal Valu As Byte)
'
Declare Sub MhOutWord Lib "MhMu32.DLL"
'          (ByVal Port As Long,
'           ByVal Valu As Integer)
'
Function PioIn$(IoChannel%)
'
Select Case IoChannel%
'
Case 0: PioIn$ = ReadPIO$(IobaseAddr% + 0, 1) 'PLC Input #0
Case 1: PioIn$ = ReadPIO$(IobaseAddr% + 0, 2) 'PLC Input #1
Case 2: PioIn$ = ReadPIO$(IobaseAddr% + 0, 4) 'PLC Input #2
Case 3: PioIn$ = ReadPIO$(IobaseAddr% + 0, 8) 'PLC Input #3
Case 4: PioIn$ = ReadPIO$(IobaseAddr% + 0, 16) 'PLC Input #4
Case 5: PioIn$ = ReadPIO$(IobaseAddr% + 0, 32) 'PLC Input #5
Case 6: PioIn$ = ReadPIO$(IobaseAddr% + 0, 64) 'PLC Input #6

```



## Module3 - 2

```

Case 7: P1oIn% = ReadPIO%(IobaseAddr% + 0, 128) 'PLC Input #7

Case 8: P1oIn% = ReadPIO%(IobaseAddr% + 1, 1) 'PLC Input #8
Case 9: P1oIn% = ReadPIO%(IobaseAddr% + 1, 2) 'PLC Input #9
Case 10: P1oIn% = ReadPIO%(IobaseAddr% + 1, 4) 'PLC Input #10
Case 11: P1oIn% = ReadPIO%(IobaseAddr% + 1, 8) 'PLC Input #11
Case 12: P1oIn% = ReadPIO%(IobaseAddr% + 1, 16) 'PLC Input #12
Case 13: P1oIn% = ReadPIO%(IobaseAddr% + 1, 32) 'PLC Input #13
Case 14: P1oIn% = ReadPIO%(IobaseAddr% + 1, 64) 'PLC Input #14
Case 15: P1oIn% = ReadPIO%(IobaseAddr% + 1, 128) 'PLC Input #15

Case 16: P1oIn% = ReadPIO%(IobaseAddr% + 2, 1) 'PLC Input #16
Case 17: P1oIn% = ReadPIO%(IobaseAddr% + 2, 2) 'PLC Input #17
Case 18: P1oIn% = ReadPIO%(IobaseAddr% + 2, 4) 'PLC Input #18
Case 19: P1oIn% = ReadPIO%(IobaseAddr% + 2, 8) 'PLC Input #19
Case 20: P1oIn% = ReadPIO%(IobaseAddr% + 2, 16) 'PLC Input #20
Case 21: P1oIn% = ReadPIO%(IobaseAddr% + 2, 32) 'PLC Input #21
Case 22: P1oIn% = ReadPIO%(IobaseAddr% + 2, 64) 'PLC Input #22
Case 23: P1oIn% = ReadPIO%(IobaseAddr% + 2, 128) 'PLC Input #23

Case 24: P1oIn% = ReadPIO%(IobaseAddr% + 4, 1) 'PLC Input #24
Case 25: P1oIn% = ReadPIO%(IobaseAddr% + 4, 2) 'PLC Input #25
Case 26: P1oIn% = ReadPIO%(IobaseAddr% + 4, 4) 'PLC Input #26
Case 27: P1oIn% = ReadPIO%(IobaseAddr% + 4, 8) 'PLC Input #27
Case 28: P1oIn% = ReadPIO%(IobaseAddr% + 4, 16) 'PLC Input #28
Case 29: P1oIn% = ReadPIO%(IobaseAddr% + 4, 32) 'PLC Input #29
Case 30: P1oIn% = ReadPIO%(IobaseAddr% + 4, 64) 'PLC Input #30
Case 31: P1oIn% = ReadPIO%(IobaseAddr% + 4, 128) 'PLC Input #31

Case 32: P1oIn% = ReadPIO%(IobaseAddr% + 5, 1) 'PLC Input #32
Case 33: P1oIn% = ReadPIO%(IobaseAddr% + 5, 2) 'PLC Input #33
Case 34: P1oIn% = ReadPIO%(IobaseAddr% + 5, 4) 'PLC Input #34
Case 35: P1oIn% = ReadPIO%(IobaseAddr% + 5, 8) 'PLC Input #35
Case 36: P1oIn% = ReadPIO%(IobaseAddr% + 5, 16) 'PLC Input #36
Case 37: P1oIn% = ReadPIO%(IobaseAddr% + 5, 32) 'PLC Input #37
Case 38: P1oIn% = ReadPIO%(IobaseAddr% + 5, 64) 'PLC Input #38
Case 39: P1oIn% = ReadPIO%(IobaseAddr% + 5, 128) 'PLC Input #39

Case 40: P1oIn% = ReadPIO%(IobaseAddr% + 6, 1) 'PLC Input #40
Case 41: P1oIn% = ReadPIO%(IobaseAddr% + 6, 2) 'PLC Input #41
Case 42: P1oIn% = ReadPIO%(IobaseAddr% + 6, 4) 'PLC Input #42
Case 43: P1oIn% = ReadPIO%(IobaseAddr% + 6, 8) 'PLC Input #43
Case 44: P1oIn% = ReadPIO%(IobaseAddr% + 6, 16) 'PLC Input #44
Case 45: P1oIn% = ReadPIO%(IobaseAddr% + 6, 32) 'PLC Input #45
Case 46: P1oIn% = ReadPIO%(IobaseAddr% + 6, 64) 'PLC Input #46
Case 47: P1oIn% = ReadPIO%(IobaseAddr% + 6, 128) 'PLC Input #47

Case 48: P1oIn% = ReadPIO%(IobaseAddr% + 8, 1) 'PLC Input #48
Case 49: P1oIn% = ReadPIO%(IobaseAddr% + 8, 2) 'PLC Input #49
Case 50: P1oIn% = ReadPIO%(IobaseAddr% + 8, 4) 'PLC Input #50
Case 51: P1oIn% = ReadPIO%(IobaseAddr% + 8, 8) 'PLC Input #51
Case 52: P1oIn% = ReadPIO%(IobaseAddr% + 8, 16) 'PLC Input #52
Case 53: P1oIn% = ReadPIO%(IobaseAddr% + 8, 32) 'PLC Input #53
Case 54: P1oIn% = ReadPIO%(IobaseAddr% + 8, 64) 'PLC Input #54
Case 55: P1oIn% = ReadPIO%(IobaseAddr% + 8, 128) 'PLC Input #55

Case 56: P1oIn% = ReadPIO%(IobaseAddr% + 9, 1) 'PLC Input #56
Case 57: P1oIn% = ReadPIO%(IobaseAddr% + 9, 2) 'PLC Input #57
Case 58: P1oIn% = ReadPIO%(IobaseAddr% + 9, 4) 'PLC Input #58
Case 59: P1oIn% = ReadPIO%(IobaseAddr% + 9, 8) 'PLC Input #59
Case 60: P1oIn% = ReadPIO%(IobaseAddr% + 9, 16) 'PLC Input #60
Case 61: P1oIn% = ReadPIO%(IobaseAddr% + 9, 32) 'PLC Input #61
Case 62: P1oIn% = ReadPIO%(IobaseAddr% + 9, 64) 'PLC Input #62
Case 63: P1oIn% = ReadPIO%(IobaseAddr% + 9, 128) 'PLC Input #63

Case 64: P1oIn% = ReadPIO%(IobaseAddr% + 10, 1) 'PLC Input #64
Case 65: P1oIn% = ReadPIO%(IobaseAddr% + 10, 2) 'PLC Input #65

```

Module3 - 3

```

Case 66: PioIn% = ReadPIO$(IobaseAddr% + 10, 4) 'PLC Input #66
Case 67: PioIn% = ReadPIO$(IobaseAddr% + 10, 8) 'PLC Input #67
Case 68: PioIn% = ReadPIO$(IobaseAddr% + 10, 16) 'PLC Input #68
Case 69: PioIn% = ReadPIO$(IobaseAddr% + 10, 32) 'PLC Input #69
Case 70: PioIn% = ReadPIO$(IobaseAddr% + 10, 64) 'PLC Input #70
Case 71: PioIn% = ReadPIO$(IobaseAddr% + 10, 128) 'PLC Input #71

```

End Select

End Function

Sub PioInit()

Dim I%

```

Call MhOutByte(IobaseAddr% + 3, &H9B) 'Set 1st 24 lines as inputs
Call MhOutByte(IobaseAddr% + 7, &H9B) 'Set 2nd 24 lines as inputs
Call MhOutByte(IobaseAddr% + 11, &H9B) 'Set 3rd 24 lines as inputs
Call MhOutByte(IobaseAddr% + 15, &H80) 'Set next 24 lines as outputs
Call MhOutByte(IobaseAddr% + 19, &H80) 'Set last 24 lines as outputs

```

```

For I% = 0 To 7
    OutImage%(I%) = 0 'Clear the image array
Next I%

```

```

Call MhOutByte(IobaseAddr% + 12, 255) 'Bit 0 - 7
Call MhOutByte(IobaseAddr% + 13, 255) 'Bit 8 - 15
Call MhOutByte(IobaseAddr% + 14, 255) 'Bit 16 - 23
Call MhOutByte(IobaseAddr% + 16, 255) 'Bit 24 - 31
Call MhOutByte(IobaseAddr% + 17, 255) 'Bit 32 - 39
Call MhOutByte(IobaseAddr% + 18, 255) 'Bit 40 - 47
'Set all hardware port bits initially to 0

```

End Sub

Sub PioOut(IoChannel%, NuVal%)

Select Case IoChannel%

```

Case 0: Call WritePIO(IobaseAddr% + 12, 0, 1, NuVal%) 'PLC Output #0
Case 1: Call WritePIO(IobaseAddr% + 12, 0, 2, NuVal%) 'PLC Output #1
Case 2: Call WritePIO(IobaseAddr% + 12, 0, 4, NuVal%) 'PLC Output #2
Case 3: Call WritePIO(IobaseAddr% + 12, 0, 8, NuVal%) 'PLC Output #3
Case 4: Call WritePIO(IobaseAddr% + 12, 0, 16, NuVal%) 'PLC Output #4
Case 5: Call WritePIO(IobaseAddr% + 12, 0, 32, NuVal%) 'PLC Output #5
Case 6: Call WritePIO(IobaseAddr% + 12, 0, 64, NuVal%) 'PLC Output #6
Case 7: Call WritePIO(IobaseAddr% + 12, 0, 128, NuVal%) 'PLC Output #7

Case 8: Call WritePIO(IobaseAddr% + 13, 1, 1, NuVal%) 'PLC Output #8
Case 9: Call WritePIO(IobaseAddr% + 13, 1, 2, NuVal%) 'PLC Output #9
Case 10: Call WritePIO(IobaseAddr% + 13, 1, 4, NuVal%) 'PLC Output #10
Case 11: Call WritePIO(IobaseAddr% + 13, 1, 8, NuVal%) 'PLC Output #11
Case 12: Call WritePIO(IobaseAddr% + 13, 1, 16, NuVal%) 'PLC Output #12
Case 13: Call WritePIO(IobaseAddr% + 13, 1, 32, NuVal%) 'PLC Output #13
Case 14: Call WritePIO(IobaseAddr% + 13, 1, 64, NuVal%) 'PLC Output #14
Case 15: Call WritePIO(IobaseAddr% + 13, 1, 128, NuVal%) 'PLC Output #15

Case 16: Call WritePIO(IobaseAddr% + 14, 2, 1, NuVal%) 'PLC Output #16
Case 17: Call WritePIO(IobaseAddr% + 14, 2, 2, NuVal%) 'PLC Output #17
Case 18: Call WritePIO(IobaseAddr% + 14, 2, 4, NuVal%) 'PLC Output #18
Case 19: Call WritePIO(IobaseAddr% + 14, 2, 8, NuVal%) 'PLC Output #19
Case 20: Call WritePIO(IobaseAddr% + 14, 2, 16, NuVal%) 'PLC Output #20
Case 21: Call WritePIO(IobaseAddr% + 14, 2, 32, NuVal%) 'PLC Output #21
Case 22: Call WritePIO(IobaseAddr% + 14, 2, 64, NuVal%) 'PLC Output #22
Case 23: Call WritePIO(IobaseAddr% + 14, 2, 128, NuVal%) 'PLC Output #23

Case 24: Call WritePIO(IobaseAddr% + 16, 3, 1, NuVal%) 'PLC Output #24

```

## Module3 - 4

```

Case 25: Call WritePIO(IobaseAddr% + 16, 3, 2, NuVal%) 'PLC Output #25
Case 26: Call WritePIO(IobaseAddr% + 16, 3, 4, NuVal%) 'PLC Output #26
Case 27: Call WritePIO(IobaseAddr% + 16, 3, 8, NuVal%) 'PLC Output #27
Case 28: Call WritePIO(IobaseAddr% + 16, 3, 16, NuVal%) 'PLC Output #28
Case 29: Call WritePIO(IobaseAddr% + 16, 3, 32, NuVal%) 'PLC Output #29
Case 30: Call WritePIO(IobaseAddr% + 16, 3, 64, NuVal%) 'PLC Output #30
Case 31: Call WritePIO(IobaseAddr% + 16, 3, 128, NuVal%) 'PLC Output #31

Case 32: Call WritePIO(IobaseAddr% + 17, 4, 1, NuVal%) 'PLC Output #32
Case 33: Call WritePIO(IobaseAddr% + 17, 4, 2, NuVal%) 'PLC Output #33
Case 34: Call WritePIO(IobaseAddr% + 17, 4, 4, NuVal%) 'PLC Output #34
Case 35: Call WritePIO(IobaseAddr% + 17, 4, 8, NuVal%) 'PLC Output #35
Case 36: Call WritePIO(IobaseAddr% + 17, 4, 16, NuVal%) 'PLC Output #36
Case 37: Call WritePIO(IobaseAddr% + 17, 4, 32, NuVal%) 'PLC Output #37
Case 38: Call WritePIO(IobaseAddr% + 17, 4, 64, NuVal%) 'PLC Output #38
Case 39: Call WritePIO(IobaseAddr% + 17, 4, 128, NuVal%) 'PLC Output #39

Case 40: Call WritePIO(IobaseAddr% + 18, 5, 1, NuVal%) 'PLC Output #40
Case 41: Call WritePIO(IobaseAddr% + 18, 5, 2, NuVal%) 'PLC Output #41
Case 42: Call WritePIO(IobaseAddr% + 18, 5, 4, NuVal%) 'PLC Output #42
Case 43: Call WritePIO(IobaseAddr% + 18, 5, 8, NuVal%) 'PLC Output #43
Case 44: Call WritePIO(IobaseAddr% + 18, 5, 16, NuVal%) 'PLC Output #44
Case 45: Call WritePIO(IobaseAddr% + 18, 5, 32, NuVal%) 'PLC Output #45
Case 46: Call WritePIO(IobaseAddr% + 18, 5, 64, NuVal%) 'PLC Output #46
Case 47: Call WritePIO(IobaseAddr% + 18, 5, 128, NuVal%) 'PLC Output #47

End Select

End Sub

Function ReadPIO%(Addr%, Pos%)

Dim I%

I% = MhInpByte(Addr%) 'Read the Hardware input
If (I% And Pos%) <> 0 Then 'Evaluate the bit
    ReadPIO% = False 'Return true if set
Else
    ReadPIO% = True 'False if clear
End If

End Function

Sub WritePIO(Addr%, ArrayPos%, BitPos%, NuVal%)

Dim I%

If NuVal% = 0 Then
    I% = OutImage%(ArrayPos%) And (Not BitPos%) 'Clear the bit if 0
Else
    I% = OutImage%(ArrayPos%) Or BitPos% 'Set the bit if 1
End If

OutImage%(ArrayPos%) = I% 'Update the image register
Call MhOutByte(Addr%, (Not (I%) And 255)) 'Update the hardware register

End Sub

Public Sub ScanPLCInputs()

Dim I%

'Routine reads all input channels and maps to PLC_Imap%. Subscript
'points to the channel number.

```

Module3 - 5

```
For I% = 0 To 71
    PLC_I_Last%(I%) = PLC_I%(I%)
    PLC_I%(I%) = PioIn%(I%) 'Read the bit
Next I%
```

End Sub

Public Sub StrobePLCOutputs()

Dim I%

'Routine writes contents of PLC\_Omap% to appropriate discrete output

For I% = 0 To 47

Call PioOut(I%, PLC\_O%(I%)) 'Write the array to discrete outputs

Next I%

End Sub

## Module2 - 1

```
'Program      : Press_00.exe
'Module       : MASTEXEC.BAS
'Creation date : 29-Sep-93
'Version      : V2.1
'Edit         : -01
'Edit date    : 26-Aug-00
'Author       : Joseph B. Schutte III
               : Industrial Light & Motion, Inc.
               : 2170 Van Blaricum Rd.
               : Cincinnati OH 45233 USA
'
' Copyright (C) 1993-2000 Industrial Light & Motion, Inc.
```

## 'Revision History

'Version/Edit	Edit date	Who	Reason
'V2.1-01	26-Aug-00	JBSIII	Add special functions for Press_00
'V2.1-00	20-Jul-97	JBSIII	Add functions to support Web Perforator
'V2.0-00	20-Feb-97	JBSIII	Major upgrade to package to support VB4 and 32 bit code
'V1.0-06	25-Jan-97	JBSIII	Change homing seq to utilize HM command rather than FE/FI.
'V1.0-05	07-Jan-97	JBSIII	Fix continuous mode bug and improve perf cycle execution
'V1.0-04	21-Dec-96	JBSIII	Add move to unload pos to ref search
'V1.0-03	06-Oct-96	JBSIII	Add programmable ref search P112
'V1.0-01	27-Aug-94	JBSIII	Add I/O diag variables
'V1.0-02	28-Aug-94	JBSIII	Create Init System to get access for powerdown

## Option Explicit

```
Global MasCmdPos!(16) 'Master desired position (0)=A
Global MasActPos!(16) 'Actual position based on current reference
Global MasToGoPos!(16) 'Remainder of distance to travel
Global MasFeedRate!(16) 'This is the feedrate in IPM
Global MasRefMode%(16) 'Reference mode, 0=ABS 1=Left 2=Right
Global LMasCmdPos!(16) 'Use these as edge detectors for UpdateAxDisplay
Global LMasActPos!(16)
Global LMasToGoPos!(16)
Global LMasFeedRate!(16)
Global LMasRefMode%(16)

Global MasCPI!(16) 'This is the counts per inch for data conversion
Global MasCPR!(16) 'Counts per rev

Global ABSposoffset!(16) 'This is the offset distance to ABS
Global CFLErrorCnt% 'Running trend counter for CF1 errors (AutoIntegrator)
Global CF1Miss% 'This is the number of contiguous missed sheeter marks
Global ConfigTrace% 'Used for returns from Npad and Apad
Global Counter_NRS% 'Master non-resetable counter
Global Counter_shift1% 'Resetable counter #1
Global Counter_shift2% 'Resetable counter #2
Global Cutoff_CPR!
Global CutoffEnable% 'Set when Cutoff unit enabled
Global DiecuttingEnable% 'Set when Diecutting Unit enabled
Global DposerError! 'Diecutter position error
Global EnableHostPolling1% 'Set to zero to disable counter/data updates
Global EnableHostPolling2% 'Same, CPU #2
Global FlexoCalibrated% 'Flexo calibrated
Global Flexo_CPR! 'Modulus value for ABS postion
Global IntaglioCalReq% 'This is the request flag from CPU-0
Global Intaglio_CPR! 'Modulus calc for ABS positions
Global Intaglio_Mod factor% 'Number of Intaglio revs in a repeat
Global IntChillrollEnable% 'Chill roll enable
```

## Module2 - 2

```

Global IntInfeedEnable% 'Product infeed enable
Global IntOutfeedEnable% 'Product outfeed enable
Global IntShuttleEnable% 'Shuttle roll enable
Global IntWipingEnable% 'Wiping servo enable
Global Iposerror! 'Intaglio position error
Global IP_Move% 'Commanded change this cycle
Global IP_offset# 'This is the IP (Intaglio) CAM offset
Global HistogramStrobe% 'Set once per CF1 rev to sync Histograms
Global HistogramApos% 'Perf histogram pointer
Global HistogramBpos% 'Sheeter histogram pointer
Global HistoLastTick# 'Histogram's last count pos
Global IntaglioCalibrated% 'Intaglio calibration status
Global LastAxis% 'Parameter load sets last active axis
Global LastCF1% '+1=CF1 err trend is +, -1=CF1 err trend is -
Global LastCodeLine2% 'This is the last line in the program
Dim LastCF1tick%
Global LastMotionCnt% 'Previous scan counter value from Galil
Global Lube_counter% 'Number of impressions since last lube cycle
Global MachineCalibrated% 'Global calibrate signal set by PLC
Global MotionCnt% 'Counter value read from Motion subsystem
Global MotionSystemStatus% 'Motion system status read from card
Global Numbering_CPR!
Global NumberingEnable% 'Set when Numbering unit enabled
Global ProductCalibrated% 'VB Loop ready on product system
Global P00Flexotrace%
Global P00Intagliotrace%
Dim Parsed_Str$(50)
Dim Parsed_Count% 'Command Parser data entry counter
Dim Parsed_Count2%
Dim Parsed_Str$(50) 'Command Parser target array
Global Pressruntrace% 'Used for returns from keypad to frmPressrun
Global PWRewindDia 'Intaglio paperwipe roll diameter sensor
Global EditActive% 'Flag used to inhibit updates when not in edit mode
Global Npad_Val! 'Result value from numeric pad touch screen
Global Apad_Text% 'Result value from ascii pad touch screen
Global RegIntaglioAuto% 'Set when Intaglio register control in auto
Global RegDiecuttingAuto% 'Set when Diecutter in Auto
Global RegNumberingAuto% 'Set when Numbering in Auto
Global RegCutoffAuto% 'Set when Cutoff in Auto register
Global Ridethrudeley% 'Motion ride through timer flag
Global SaveOpControls% 'Image of operator PB status
Global SHIMiss% 'Counter for missed register marks
Global PressRollData$(20) 'Holding location for press production data
Global RollDataLoaded% 'Flag that is set when valid production data loaded
Global OpPositionOffset# 'this is the operator running bias
Global SaveRollEnable% 'Enables regular updates of roll data to disk
Global ScanOpPb% 'Used for edge detection in operator controls
Global Startdelay% 'Motion start delay timer flag
Global SysMode% 'Machine mode: 1=Flexo, 2= Intaglio, 3=Combination
Global SystemTickCounter# 'Number of impressions since bootup
Global System_circumference# 'This is the current system circumference
Global TargetPos# 'Work variable used as target position for transport moves
Global Setuptrace% 'Returns from frmConfig
Global Paramstrace% 'Returns from frmPtable
Global Productiontrace% 'Returns from frmProduction
Global Testvar% 'Temp variable for testing system
Global WhoseFocus% '1=Pressrun, 2=Flexo, 3=Params, 4=Production
                  '5=Config, 6=Intaglio
Global WipingCalibrated% 'VB Loop ready on wiping system

Global LastSystemState% 'This is the previous message in the status box

Global LastMasterState%

Global SystemStatus% 'Main System Status word
                  'Bit definitions:
                  '0 -

```

Module2 - 3

```
'1 -
'2 -
'3 -
'4 -
'5 -
'6 -
'7 -
'8 - Power ON status 1=On
'9 -
'10 -
'11 -
'12 -
'13 -
'14 -
'15 -
```

```
Global VarChangeFlag%(100) 'Flags set when operator data changes
                             'Definition table:
'0 - Intaglio Register offset
'1 - Diecutting Register offset
'2 - Numbering Register offset
'3 - Cutoff Register offset
'4 - System mode changed
'5 - Intaglio register auto/manual
'6 - Diecutting register auto/manual
'7 - Numbering register auto/manual
'8 - Cutoff register auto/manual
'9 - Line speed change
'10- Numbering repeat length change
'11- Cutoff repeat length change
'12- Flexo gearing bias
'13- Product Infeed bias
'14- SH1 gearing bias
'15- SH2 gearing bias
'16- Product outfeed bias
'17- Wiping Infeed bias
'18- Wiping Outfeed bias
'19- Chill rolls gearing bias
'20- Diecutting gearing bias
'21- Numbering gearing bias
'22- Folder/Cutoff gearing bias
'23- Wiping speed trim ratio change
'24- Product Calibration request
'25- Wiping Calibration request
'26- Diecutting Online status changed
'27- Numbering Online status changed
'28- Cutoff Inline status changed
'29- System circumference changed
'30- Inker #1 Auto speed change
'31- Inker #1 Manual speed change
'32- Inker #2 Auto speed change
'33- Inker #2 Manual speed change
'34- Inker #1 mode
'35- Inker #2 mode
'36- Intaglio Impression mode
'37- Intaglio Wiping mode
'38- Intaglio Prewipe mode
'39- Flexo Impression mode changed
'40- Wiping System mode
'41- Request to recalc Cam table
'42- Making a change to IP offset
'43- Flexo register offset changed
'44- Intaglio lower window changed
'45- Intaglio upper window changed
'46- Diecutter lower window changed
'47- Diecutter upper window changed
'48- Cutoff lower window changed
```

Module2 - 4

'49- Cutoff upper window changed  
 '50- Forces a clear on the product calibrate flag

```

Global IoInput_tbl$(10, 8) 'I/O diagnostic input table
Global IoOutput_tbl$(10, 8) 'I/O diagnostic output table
Global IoInput_tblu$(10, 8) 'I/O diagnostic input updates
Global IoOutput_tblu$(10, 8) 'I/O diagnostic output updates

Public Sub Update_production_screen()

    FrmProduction.MhRealInput9.Text = Str$(SystemTickCount#) 'Counter_NRS&
    FrmProduction.MhRealInput11.Text = Str$(Counter_shift1&)
    FrmProduction.Text5.Text = Str$(ActualLinSpeed%)

End Sub

Function CmdParser$(SrcStr$, Dlm$)

    Dim I%, J%, K%

    K% = 0
    I% = 0
    J% = InStr(2, SrcStr$, Dlm$)

    While J% > 0
        Parsed_Str$(K%) = Mid$(SrcStr$, I% + 1, J% - I% - 1)
        I% = J%
        J% = InStr(I% + 1, SrcStr$, Dlm$)
        K% = K% + 1
    Wend

    Parsed_Str$(K%) = Mid$(SrcStr$, I% + 1, 40) 'Get the last entry
    K% = K% + 1

    Parsed_Str$(K%) = "" 'This is the END flag

    Parsed_Count% = K%
    CmdParser% = K%

End Function

Sub Init_System()

    Dim I%, J%
    Call LoadParameters 'Load the parameters from disk
    Call LoadProductiondata 'Load the production counters
    Call PioInit 'Initialize the Parallel I/O
    Call AioInit 'Initialize the Analog output board
    IntaglioCalReq% = 0
    IP_offset# = 0
    frmConfig!MhRealInput6.Text = 0
    LastCF1tick% = 0 'Used for generating once per rev tick
    SystemStatus% = 0 'Initialize all state bits to 0
    GuardsBypassed% = 0
    Pressruntrace% = 0
    P00Flexotrace% = 0
    P00Intagliotrace% = 0
    Setuptrace% = 0
    Paramstrace% = 0
    Lube_counter% = 0
    CF1errorcnt% = 0
    LastCF1% = 0
    LinSpeedIntegrator! = 0
    HistoLastTick# = 0
    DiecuttingEnable% = 1

```



Module2 - 5

```

NumberingEnable% = 1
Ridethrudelay% = 0
Startdelay% = 0

frmConfig!MhRealInput7.Text = 23# 'Temporary value for init
frmConfig!MhRealInput8.Text = 1# 'Temp value
System_circumference# = frmConfig!MhRealInput7.Text
'Initialize Job data variables here

I% = InitServo%() 'Init Servo board(s)
If I% <> 0 Then 'Fatal initialization error if <> 0
    Call PioOut(0, 0)
    MsgBox ("?System initialization failed")
End If

I% = NewJobData% 'Load job parameters
' If Len(FrmProduction!Text8.Text) > 3 Then
'     Call LoadRollData
' End If
frmConfig!MhRealInput7.Text = "23.0"
frmConfig!MhRealInput8.Text = Param_tbl$(1332, 2)
frmConfig!MhRealInput9.Text = Param_tbl$(1432, 2)
'Get initial values from Param table

For I% = 0 To 50
    VarChangeFlag%(I%) = 1
Next I%
Call UpdateMotionSystemVars
'Set all flags true and call updater

LastMasterState% = 0 'Clear last state data
EditActive% = False 'Not initially .In Edit mode

I% = CmdServo1$("ParamOK=1") 'Must be last!!! - Enables motion prog
I% = CmdServo2$("ParamOK=1")

frmMain!Timer2.Enabled = True 'Enable System logic processor
frmMain!Timer3.Enabled = True 'Enable screen updates
frmMain!Timer4.Enabled = True 'Enable long term disk data updates
frmMain!Timer5.Enabled = True 'Activate the PLC processor
EnableHostPolling1% = True 'Enable Motion variable reads
EnableHostPolling2% = True

End Sub

Sub Calculate_Flexo_modulus()

Dim A, I%, J, K

'This routine calculates Master Modulus
A = 0.5
J = frmConfig!MhRealInput7.Text 'Intaglio repeat
K = Param_tbl$(1132, 2) 'Get the Flexo/Diecutting repeat

I% = 0
While Int(A) <> A 'Loop till A is an integer
    I% = I% + 1 'This is the intaglio modulus
    A = (J / K) * I% 'Flexo modulus
Wend

Intaglio_Mod_factor% = I%
Intaglio_CPR! = Param_tbl$(802, 2) * I%
Flexo_CPR! = Param_tbl$(1102, 2) * A

End Sub

```

Module2 - 6

```

Function LoadJobdata%()
    ' This module loads the data from the CNC files

    Dim I$, J$, K$, L$

    On Error GoTo Loadfail

    I$ = Param_tbl$(0, 2) & Trim$(frmMain!Text6.Text) & ".JOB"
    Open I$ For Input As 1

    Input #1, I$ 'Read the creation date
    frmP00Flexo!Text1.Text = I$

'ADD CODE HERE TO LOAD INIT

    LoadJobdata% = 0 'Successful load
    SystemStatus% = SystemStatus% Or &H20 'Set FILE LOADED bit
    Close #1

    'Force update on vars to motion system

LoadJobdata_1:
    Exit Function

Loadfail:
    MsgBox ("Error " & Err & " reading Job file")
    LoadJobdata% = -1
    Close #1
    Resume LoadJobdata_1

End Function

Sub LoadParameters()
    On Error GoTo LoadTrap

    Dim I$, J$, K$, L$

    Open "c:\Press00\params.sys" For Input As 1

    For I$ = 1 To 2000 'Setup the loop for the load
        frmPtable.MSFlexGrid1.Row = I$ - 1
        frmPtable.MSFlexGrid1.Col = 0
        frmPtable.MSFlexGrid1.Text = Str$(I$)
        Line Input #1, L$ 'Get a Line from file
        J$ = InStr(1, L$, ";")
        K$ = InStr(J$ + 1, L$, ";")
        frmPtable.MSFlexGrid1.Col = 1
        frmPtable.MSFlexGrid1.Text = Mid$(L$, J$ + 1, K$ - J$ - 1)
        frmPtable.MSFlexGrid1.Col = 2
        frmPtable.MSFlexGrid1.Text = Mid$(L$, K$ + 1, 20)
    Next I$

    LastAxis% = Cint(Param_tbl$(10, 2)) - 1

LExit:
    Close 1 'Close the parameter file
    Exit Sub

LoadTrap:

```

Module2 - 7

Resume LExit

End Sub

Sub LoadProductiondata()

'This procedure reads the disk based storage to load counters  
'on powerup

Dim I\$

Open "C:\Press00\Press\_00.SYS" For Input As 1

Input #1, I\$ 'This is the Non-reset sheet totalizer  
Counter\_NRS& = CDb1(I\$)  
FrmProduction!MhRealInput9.Text = I\$

Input #1, I\$ 'This is the shift sheet (resetable) count  
Counter\_shift1& = CDb1(I\$)  
FrmProduction!MhRealInput11.Text = I\$

Input #1, I\$ 'This is the operator ID  
FrmProduction!Text1.Text = I\$

Input #1, I\$ 'Read the last used roll data file  
FrmProduction!Text8.Text = I\$

Close #1

End Sub

Function LoadRollData%( )

' This module loads the data from the SN files

Dim I\$, J\$, K%

On Error GoTo Rollfail

I\$ = FrmProduction!Text8.Text & ".DAT"  
Open I\$ For Input As 1

Line Input #1, J\$ 'Read the roll identifier  
FrmProduction!Text4.Text = J\$

For K% = 1 To 10 'Read the Press specific data  
Input #1, PressRollData\$(K%)  
Next K%

Line Input #1, J\$ 'Read Operator A  
FrmProduction!Text6.Text = J\$

Line Input #1, J\$ 'Read Operator B  
FrmProduction!Text7.Text = J\$

Line Input #1, J\$ 'Read Total impressions  
FrmProduction!MhRealInput1 = J\$

Line Input #1, J\$ 'Read Good impressions  
FrmProduction!MhRealInput2 = J\$

Line Input #1, J\$ 'Read Printing spoils  
FrmProduction!MhRealInput3 = J\$

Line Input #1, J\$ 'Read Diecutting register spoils

Module2 - 8

```

FrmProduction!MhRealInput4 = J$

Line Input #1, J$ 'Read Numbering spoils
FrmProduction!MhRealInput5 = J$

Line Input #1, J$ 'Read Sheeter register spoils
FrmProduction!MhRealInput6 = J$

Line Input #1, J$ 'Read Operator forced spoils
FrmProduction!MhRealInput7 = J$

Line Input #1, J$ 'Read Total spoils
FrmProduction!MhRealInput8 = J$

Close 1
LoadRollData% = 0
RollDataLoaded% = 1 'This is a perforator RUN permissive
SaveRollEnable% = 1 'Enable auto save function

```

```

LoadRollData_1:
Exit Function

```

```

Rollfail:
MsgBox ("?Error " & Err & " reading Roll data file")
Stop
LoadRollData% = -1
Resume LoadRollData_1

```

```

End Function

```

```

Function ModString$(Sval$, Incval#, Dp%)

```

```

'This procedure either increases or decreases the numeric
'value of the source string. Result string is returned

```

```

Dim I%, J$, K#, L%

```

```

K# = Cdbl(Sval$) + Incval# 'This is the new value
J$ = Trim$(Str$(K#))
L% = InStr(1, J$, ".") 'Find the DP
ModString$ = Left$(J$, L% + Dp%) 'Truncate the string

```

```

End Function

```

```

Public Sub UpdateBargraphs()

```

```

'This routine uses Intaglioerror and CFlerror to set the bargraphs.
'Data is scaled in inches for a bargraph range of +/-1.200 inches.

```

```

Dim I!

```

```

frmPressrun!MhRealInput7.Text = Iposerror
frmPressrun!MhRealInput8.Text = Dposerror

```

```

'Update the Intaglio bar graph

```

```

If Iposerror! < 0 Then
    'It's a negative value
    frmPressrun.Mh3dGauge2.FillValue = 0
    'Clear the positive side of the bargraph
    I! = Iposerror! * -1000
    If I! >= frmPressrun.Mh3dGauge1.Max Then
        I! = frmPressrun.Mh3dGauge1.Max
    End If
    frmPressrun.Mh3dGauge1.FillValue = I!
End If
If Iposerror! >= 0 Then
    'It's a positive value

```

Module2 - 9

```

    frmPressrun.Mh3dGauge1.FillValue = 0
    I! = Iposerror! * 1000
    If I! >= frmPressrun.Mh3dGauge2.Max Then
        I! = frmPressrun.Mh3dGauge2.Max
    End If
    frmPressrun.Mh3dGauge2.FillValue = I!
End If

```

```

'Update the Diecutting bar graph
If Dposerror! < 0 Then
    'It's a negative value
    frmPressrun.Mh3dGauge4.FillValue = 0
    'Clear the positive side of the bargraph
    I! = Dposerror! * -1000
    If I! >= frmPressrun.Mh3dGauge3.Max Then
        I! = frmPressrun.Mh3dGauge3.Max
    End If
    frmPressrun.Mh3dGauge3.FillValue = I!
End If
If Dposerror! >= 0 Then
    'It's a positive value
    frmPressrun.Mh3dGauge3.FillValue = 0
    I! = Dposerror! * 1000
    If I! >= frmPressrun.Mh3dGauge4.Max Then
        I! = frmPressrun.Mh3dGauge4.Max
    End If
    frmPressrun.Mh3dGauge4.FillValue = I!
End If

```

End Sub

Function NewJobData%()

```

    ' This module initializes job variables as new files

```

```

    Dim I$, J$, K$, L$

```

```

    On Error GoTo Newfail

```

```

    If Len(frmMain!Text6.Text) = 0 Then
        frmMain!Text6.Text = "Noname"
    End If

```

```

    I$ = Param_tbl$(0, 2) & frmMain!Text6.Text & ".JOB"

```

```

    Open I$ For Output As 1 'Test open the temporary file
    Close #1

```

```

    NewJobData% = 0 'Successful load
    SystemStatus% = SystemStatus% Or &H20 'Set FILE LOADED bit

```

```

NewJobdata_1:
    Exit Function

```

```

Newfail:
    MsgBox ("?Error " & Err & " creating Job file")
    NewJobData% = -1
    Resume NewJobdata_1

```

End Function

Function SaveJobdata%()

```

    ' This module saves data to the Job data files

```

Module2 - 10

```

Dim I$, J%, K%, L%

On Error GoTo Savefail

I$ = Param_tbl$(0, 2) & frmMain!Text6.Text & ".JOB"
Open I$ For Output As 1

I$ = Date$ 'Get the current system date
Print #1, I$
frmP00Flexo!Text1.Text = I$

Close #1

SaveJobdata% = 0 'Successful load

SaveJobdata_1:
Exit Function

Savefail:
MsgBox ("?Error " & Err & " writing Job file")
SaveJobdata% = -1
Close #1
Resume SaveJobdata_1

End Function

Function SaveRolldata%()

' This module saves data to the Job data files

Dim I$, J%, K%, L%

If SaveRollEnable% = 0 Then 'Only if something to save
On Error GoTo SaveRollfail

I$ = FrmProduction!Text8.Text & ".DAT"
Open I$ For Output As 1

Print #1, FrmProduction!Text4.Text 'Save the roll identifier

For K% = 1 To 10 'Resave the Press specific data
Print #1, PressRollData$(K%)
Next K%

Print #1, FrmProduction!Text6.Text 'Save Operator A
Print #1, FrmProduction!Text7.Text 'Save Operator B
Print #1, FrmProduction!MhRealInput1 'Save Total Impressions
Print #1, FrmProduction!MhRealInput2 'Save Good impressions
Print #1, FrmProduction!MhRealInput3 'Save Printing spoils
Print #1, FrmProduction!MhRealInput4 'Save Diecutting register spoils
Print #1, FrmProduction!MhRealInput5 'Save Numbering spoils
Print #1, FrmProduction!MhRealInput6 'Save Sheeter spoils
Print #1, FrmProduction!MhRealInput7 'Save Operator forced spoils
Print #1, FrmProduction!MhRealInput8 'Save Total spoils

Close #1

SaveRolldata% = 0 'Successful save

End If

SaveRolldata_1:
Exit Function

SaveRollfail:

```

Module2 - 11

```

MsgBox ("?Error " & Err & " writing Roll data file")
SaveRolldata% = -1
Resume SaveRolldata_1

```

End Function

Function AutoExec%()

```

'This is the Top Level Auto Execution Unit. It calls routines
'to read motion system variables, makes calculations, updates
'system variables, and writes new variable data to the motion
'system.

```

Dim I%

```

Call ReadMotionData 'Read the motion system variables from card
Call UpdateSystemVariables 'Calc speeds, errors, etc.
Call AutoIntegrator 'Update CF1 gear ratio trim
Call UpdateMotionSystemRL 'Update run logic commands to motion card
Call UpdateIntaglioDrive 'Update logic and velocity commands
Call UpdateWipingDrive 'Update Wiping logic and velocity commands
Call UpdateMotionSystemVars 'Update motion system ratios, etc.
Call UpdateIntaglioInkers 'Update the velocity calc's for inkers
Call WriteMotionData 'Write new variable data to the motion controllers

```

End Function

Sub UpdateIntaglioInkers()

```

'Update the Velocity commands for both Intaglio Inker drives

```

Dim I%, J%, T\_actual%

Select Case IntaglioInker1Control% 'Inker #1

Case 0 'Inker OFF

```

    Call AioOut(3, Val(Param_tbl$(916, 2))) 'Write the null offset

```

Case 1 'Inker in Manual

```

    If IntaglioInker1Manrun% = 0 Then

```

```

        Call AioOut(3, Val(Param_tbl$(916, 2))) 'Write the null offset

```

```

    Else

```

```

        I% = Val(Param_tbl$(916, 2)) + (Val(Param_tbl$(917, 2)) * Val(frmP00Intaglio!Slider1.Val
ue))

```

```

        Call AioOut(3, I%)

```

```

    End If

```

Case 2 'Inker in Auto

```

    T_actual% = ActualLinSpeed% * 2 * (frmP00Intaglio!Slider2.Value / 100)

```

```

    'Plug in the auto mode scaling

```

```

    I% = Param_tbl$(916, 2) + (Param_tbl$(917, 2) * T_actual%)

```

```

    J% = Param_tbl$(916, 2) + (Param_tbl$(917, 2) * Param_tbl$(918, 2))

```

```

    If Abs(J%) > Abs(I%) Then I% = J% 'Run at min speed or better

```

```

    Call AioOut(3, I%)

```

End Select

Select Case IntaglioInker2Control% 'Inker #2

Case 0 'Inker OFF

```

    Call AioOut(4, Val(Param_tbl$(919, 2))) 'Write the null offset

```

Case 1 'Inker in Manual

```

    If IntaglioInker2Manrun% = 0 Then

```

```

        Call AioOut(4, Val(Param_tbl$(919, 2))) 'Write the null offset

```

```

    Else

```

```

        I% = Val(Param_tbl$(919, 2)) + (Val(Param_tbl$(920, 2)) * Val(frmP00Intaglio!Slider4.Val
ue))

```

```

        Call AioOut(4, I%)

```

Module2 - 12

```

    End If
Case 2 'Inker in Auto
    T_actual% = ActualLinSpeed% * 2 * (frmP00Intaglio!Slider3.Value / 100)
    'Plug in the auto mode scaling
    I% = Param_tbl$(919, 2) + (Param_tbl$(920, 2) * T_actual%)
    J% = Param_tbl$(919, 2) + (Param_tbl$(920, 2) * Param_tbl$(921, 2))
    If Abs(J%) > Abs(I%) Then I% = J% 'Run at min speed or better
    Call AioOut(4, I%)
End Select

End Sub
Sub WriteCounters()

    'Update the count values to disk for non-volatile

    Open "C:\Press00\Press_00.sys" For Output As #2

    Print #2, Trim$(Str$(Counter_NRS&)) 'Non-reset totalizer
    Print #2, Trim$(Str$(Counter_shift1&)) 'Shift sheet totalizer
    Print #2, frmProduction!Text1.Text 'Operator ID
    Print #2, frmProduction!Text8.Text 'Current Roll ID

    Close #2

End Sub

Public Function Param_tbl$(R%, C%)

    ' This function translates calls from the old Grid function
    ' to the new Grid function

    frmPtable.MSFlexGrid1.Row = R%
    frmPtable.MSFlexGrid1.Col = C%

    Param_tbl$ = frmPtable.MSFlexGrid1.Text

End Function

Public Sub Update_pressrun_screen()

    'This routine handles updates of all variables on the pressrun
    'display. It is the responsibility of other subs to keep the
    'variables up to date.

    If SH1MMiss% > 10 Then
        frmPressrun!Label27.Visible = True
    Else
        frmPressrun!Label27.Visible = False
    End If
    If CF1MMiss% > 10 Then
        frmPressrun!Label5.Visible = True
    Else
        frmPressrun!Label5.Visible = False
    End If
    'Update the missing mark indicators on Pressrun screen

    Call UpdateBargraphs
    frmP00Flexo!Text5.Text = ActualLinSpeed% 'Update the display

End Sub

Public Sub WriteMotionData()

    'This routine writes variable data out to the appropriate motion controllers

    Dim I%

```



Module2 - 13

```

If (EnableHostPolling1% = True) And (EnableHostPolling2 = True) Then
    I% = CmdServo1%("WipeRefS=" & Str$(ActualWipingSpeed% * System_circumference# / 49))
End If

```

End Sub

Public Sub ReadMotionData()

```

' This routine reads pertinent variables from motion card and maps
' them to the appropriate variables in the appropriate scale

```

Dim I\$, J\$, K\$, L\$, M!, N!

LastMotionCnt% = MotionCnt%

```

If (EnableHostPolling1% = True) And (EnableHostPolling2 = True) Then

```

```

    I$ = ReqServo$("MG FlexCreq", 0) 'Read the Flexo Cal req status
    K% = CmdServo2%("FlexCreq=" & Trim$(I$)) 'Map to CPU-2

```

```

    I$ = ReqServo$("MG DieCreq", 0) 'Read the Diecutting Cal req status
    K% = CmdServo2%("DieCreq=" & Trim$(I$)) 'Map to CPU-2

```

```

    I$ = ReqServo$("MG IntCreq", 0) 'Read the Intaglio Cal req status
    IntaglioCalReq% = Val(I$)

```

```

    I$ = ReqServo$("MG ALspeed", 1) 'Get the Actual Line speed
    ActualLinSpeed% = Val(I$)
    K% = CmdServo1%("ALspeed=" & I$)

```

```

    I$ = ReqServo$("MG MSIJreq", 0) 'Get the Intaglio jog request flag
    MSIntaglioJogReq% = Val(I$)

```

```

    I$ = ReqServo$("MG MSWJreq", 0) 'Get the Wiping jog request flag
    MSWipingJogReq% = Val(I$)

```

```

    I$ = ReqServo$("MG MSWJreq", 0) 'Get the Wiping imps request flag
    MSWipingImpReq% = Val(I$)

```

```

    I$ = ReqServo$("MG ProdCal", 0) 'Get the Product calibration status
    ProductCalibrated% = Val(I$)

```

```

    I$ = ReqServo$("MG WipeCal", 0) 'Get the Wiping calibration status
    WipingCalibrated% = Val(I$)

```

```

    I$ = ReqServo$("MG RevCTR", 0) 'Get the Wiping calibration status
    SystemTickCounter# = Val(I$)

```

```

    I$ = ReqServo$("MG Iposerr", 0) 'Read Intaglio position error
    Iposerror! = Cdbl(I$) / MasCPI!(7) 'Convert to inches by H axis scale

```

```

    I$ = ReqServo$("MG Dposerr", 0) 'Read Diecutter position error
    Dposerror! = Cdbl(I$) / MasCPI!(7) 'Convert to inches by H axis scale

```

```

    I$ = ReqServo$("MG FlexPerr", 1) 'Read Flexo position error
    frmPressrun!MhRealInput11.Text = Cdbl(I$)

```

```

    I$ = ReqServo$("MG NumPerr", 1) 'Read Numbering position error
    frmPressrun!MhRealInput10.Text = Cdbl(I$)

```

```

    I$ = ReqServo$("MG CutPerr", 1) 'Read Cutoff position error
    frmPressrun!MhRealInput9.Text = Cdbl(I$)

```

```

    I$ = ReqServo$("MG _TPH", 0) 'Get the IPC position
    frmConfig!MhRealInput2.Text = I$

```

Module2 - 14

```
I$ = ReqServo$("MG VBlsampo", 0) 'Get the Intaglio Mark position
frmConfig!MhRealInput11.Text = I$
```

```
I$ = ReqServo$("MG DieMKpos", 0) 'Get the Diecutter Mark position
frmConfig!MhRealInput13.Text = I$
```

```
I$ = ReqServo$("MG Cregrefm", 1) 'Get the Cutoff Mark position
frmConfig!MhRealInput14.Text = I$
```

```
I$ = ReqServo$("MG @AN[1]", 0) 'Get PW roll dia data
PWRewindDia = CSng(I$)
```

```
End If
```

```
End Sub
```

```
Public Sub UpdateSystemVariables()
```

```
'This routine called by AutoExec to update variable data and
'counters for general display update.
```

```
Dim I&
```

```

' If SystemRevTick <> 0 Then
'   I& = MotionCnt& - LastMotionCnt&
'   Counter_NRS& = Counter_NRS& + I& 'Increment Master Non-reset counter
'   Counter_shiftI& = Counter_shiftI& + I& 'Increment Shift totalizer
' End If
```

```
End Sub
```

```
Public Sub UpdateMotionSystemVars()
```

```
'This routine downloads any motion system variables that have changed
'during the current scan
```

```
Dim I%, J%, K%, L%, M$, N, O#
```

```

If VarChangeFlag%(0) <> 0 Then 'Intaglio register offset
  VarChangeFlag%(0) = 0
  I% = CmdServo1$("IntRoffs=" & Trim$(frmPressrun!MhRealInput3.Text))
End If
```

```

If VarChangeFlag%(1) <> 0 Then 'Diecutting register offset
  VarChangeFlag%(1) = 0
  I% = CmdServo1$("Dieoffs=" & Trim$(frmPressrun!MhRealInput6.Text))
  I% = CmdServo2$("Dieoffs=" & Trim$(frmPressrun!MhRealInput6.Text))
End If
```

```

If VarChangeFlag%(2) <> 0 Then 'Numbering register offset
  VarChangeFlag%(2) = 0
  I% = CmdServo2$("NumRoffs=" & Trim$(frmPressrun!MhRealInput2.Text))
End If
```

```

If VarChangeFlag%(3) <> 0 Then 'Cutoff register change
  VarChangeFlag%(3) = 0
  I% = CmdServo2$("CutRoffs=" & Trim$(frmPressrun!MhRealInput5.Text))
End If
```

```

If VarChangeFlag%(4) <> 0 Then 'System mode change
  VarChangeFlag%(4) = 0
  I% = CmdServo1$("SysMode=" & Trim$(Str$(SysMode%)))
  I% = CmdServo2$("SysMode=" & Trim$(Str$(SysMode%)))
  I% = CmdServo1$("PIEna=" & Trim$(Str$(IntInfeedEnable%)))
  I% = CmdServo1$("SHEna=" & Trim$(Str$(IntShuttleEnable%)))
  I% = CmdServo1$("POEna=" & Trim$(Str$(IntOutfeedEnable%)))
  I% = CmdServo1$("ChiEna=" & Trim$(Str$(IntChillrollEnable%)))
```

Module2 - 15

```

I% = CmdServo1%("WIEna=" & Trim$(Str$(IntWipingEnable%)))
I% = CmdServo1%("SysRun=10")
I% = CmdServo2%("SysRun=10")
End If

If VarChangeFlag%(5) <> 0 Then 'Intaglio Auto/manual status
  VarChangeFlag%(5) = 0
  I% = CmdServo1%("IntRauto=" & Trim$(Str$(RegIntaglioAuto%)))
End If

If VarChangeFlag%(6) <> 0 Then 'Diecutting Auto/manual status
  VarChangeFlag%(6) = 0
  I% = CmdServo1%("DieRauto=" & Trim$(Str$(RegDiecuttingAuto%)))
End If

If VarChangeFlag%(7) <> 0 Then 'Numbering Auto/manual status
  VarChangeFlag%(7) = 0
  I% = CmdServo2%("NumRauto=" & Trim$(Str$(RegNumberingAuto%)))
End If

If VarChangeFlag%(8) <> 0 Then 'Cutoff Auto/manual status
  VarChangeFlag%(8) = 0
  I% = CmdServo2%("CutRauto=" & Trim$(Str$(RegCutoffAuto%)))
End If

If VarChangeFlag%(9) <> 0 Then 'Line speed change
  VarChangeFlag%(9) = 0
  I% = CmdServo1%("LinSpeed=" & Trim$(frmPressrun!Text2.Text))
  I% = CmdServo2%("LinSpeed=" & Trim$(frmPressrun!Text2.Text))
End If

VarChangeFlag%(10) = 0 ' ***** Function Disabled *****
If VarChangeFlag%(10) <> 0 Then 'Numbering repeat length
  VarChangeFlag%(10) = 0
  N = System_circumference# * Intaglio_Mod_factor% / frmConfig!MhRealInput8.Text
  I% = CmdServo2%("CPRK=" & Trim$(Str$(N * Param_tbl$(1302, 2))))
  I% = CmdServo2%("CtWghtK=" & Trim$(Param_tbl$(1302, 2) / frmConfig!MhRealInput8.Text))
End If

VarChangeFlag%(11) = 0 ' ***** Function Disabled *****
If VarChangeFlag%(11) <> 0 Then 'Cutoff repeat length
  VarChangeFlag%(11) = 0
  N = System_circumference# * Intaglio_Mod_factor% / frmConfig!MhRealInput9.Text
  I% = CmdServo2%("CPRL=" & Trim$(Str$(N * Param_tbl$(1402, 2))))
  I% = CmdServo2%("CtWghtL=" & Trim$(Param_tbl$(1402, 2) / frmConfig!MhRealInput9.Text))
End If

If VarChangeFlag%(12) <> 0 Then 'Flexo gearing bias
  VarChangeFlag%(12) = 0
  I% = CmdServo2%("BiasA=" & frmPressrun!MhRealInput1(0).Text)
End If

If VarChangeFlag%(13) <> 0 Then 'Product infeed gearing bias
  VarChangeFlag%(13) = 0
  I% = CmdServo1%("BiasA=" & frmPressrun!MhRealInput1(1).Text)
End If

If VarChangeFlag%(14) <> 0 Then 'SH1 gearing bias
  VarChangeFlag%(14) = 0
  I% = CmdServo1%("BiasB=" & frmPressrun!MhRealInput1(2).Text)
End If

If VarChangeFlag%(15) <> 0 Then 'SH2 gearing bias
  VarChangeFlag%(15) = 0
  I% = CmdServo1%("BiasC=" & frmPressrun!MhRealInput1(3).Text)
End If

```

Module2 - 16

```

If VarChangeFlag%(16) <> 0 Then 'Product outfeed gearing bias
  VarChangeFlag%(16) = 0
  I% = CmdServo1%("BiasD=" & frmPressrun!MhRealInput1(4).Text)
End If

If VarChangeFlag%(17) <> 0 Then 'Wiping Infeed gearing bias
  VarChangeFlag%(17) = 0
  I% = CmdServo1%("BiasE=" & frmPressrun!MhRealInput1(5).Text)
End If

If VarChangeFlag%(18) <> 0 Then 'Wiping Outfeed gearing bias
  VarChangeFlag%(18) = 0
  I% = CmdServo1%("BiasF=" & frmPressrun!MhRealInput1(6).Text)
End If

If VarChangeFlag%(19) <> 0 Then 'Chill rolls gearing bias
  VarChangeFlag%(19) = 0
  I% = CmdServo1%("BiasG=" & frmPressrun!MhRealInput1(7).Text)
End If

If VarChangeFlag%(20) <> 0 Then 'Diecutting gearing bias
  VarChangeFlag%(20) = 0
  I% = CmdServo2%("BiasB=" & frmPressrun!MhRealInput1(8).Text)
End If

If VarChangeFlag%(21) <> 0 Then 'Numbering gearing bias
  VarChangeFlag%(21) = 0
  I% = CmdServo2%("BiasC=" & frmPressrun!MhRealInput1(9).Text)
End If

If VarChangeFlag%(22) <> 0 Then 'Sheeter gearing bias
  VarChangeFlag%(22) = 0
  I% = CmdServo2%("BiasD=" & frmPressrun!MhRealInput1(10).Text)
End If

If VarChangeFlag%(23) <> 0 Then 'Wiping speed trim change
  VarChangeFlag%(23) = 0
  I% = CmdServo1%("WipeTrim=" & frmP00Intaglio!Slider5.Value)
End If

If VarChangeFlag%(24) <> 0 Then 'Product calibration request
  VarChangeFlag%(24) = 0
  I% = CmdServo1%("??????????")
End If

If VarChangeFlag%(25) <> 0 Then 'Wiping calibration request
  VarChangeFlag%(25) = 0
  I% = CmdServo1%("??????????")
End If

If VarChangeFlag%(26) <> 0 Then 'Diecutting online changed
  VarChangeFlag%(26) = 0
  I% = CmdServo2%("DieEna=" & Trim$(Str$(DiecuttingEnable%)))
  I% = CmdServo2%("SysRun=10")
End If

If VarChangeFlag%(27) <> 0 Then 'Numbering online changed
  VarChangeFlag%(27) = 0
  I% = CmdServo2%("NumEna=" & Trim$(Str$(NumberingEnable%)))
  I% = CmdServo2%("SysRun=10")
End If

If VarChangeFlag%(28) <> 0 Then 'Cutoff online changed
  VarChangeFlag%(28) = 0
  I% = CmdServo2%("CutEna=" & Trim$(Str$(CutoffEnable%)))
  I% = CmdServo2%("SysRun=10")
End If

```

Module2 - 17

```

If VarChangeFlag%(29) <> 0 Then 'System Circumference changed
  VarChangeFlag%(29) = 0
  System_circumference# = frmConfig!MhRealInput7.Text
  frmConfig!Label2.Visible = False 'Clear Cam table light
  Call Calculate_Flexo_modulus
  I% = CmdServo1%("SysCirc=" & Trim$(Str$(System_circumference#)))
  I% = CmdServo2%("SysCirc=" & Trim$(Str$(System_circumference#)))
  I% = CmdServo1%("Mictwght=" & Trim$(Str$(MasCPR!(7) / System_circumference#)))
  I% = CmdServo2%("Mictwght=" & Trim$(Str$(MasCPR!(7) / System_circumference#)))
  I% = CmdServo2%("CPRDX=" & Trim$(Format$(Intaglio_CPR!, "#####")))
  I% = CmdServo2%("CPRI=" & Trim$(Format$(Flexo_CPRI, "#####")))
  O# = Param_tbl$(1202, 2) * (System_circumference# / Param_tbl$(1232, 2))
  I% = CmdServo2%("CPRJ=" & Trim$(Format$(O#, "#####")))
  'VarChangeFlag%(10) = 1 'Trigger recalc of Numbering and Cutoff
  'VarChangeFlag%(11) = 1
  I% = CmdServo1%("SysRun=10")
  I% = CmdServo2%("SysRun=10")
End If

If VarChangeFlag%(30) <> 0 Then 'Inker #1 Auto speed change
  VarChangeFlag%(30) = 0
  I% = CmdServo1%("????????????")
End If

If VarChangeFlag%(31) <> 0 Then 'Inker #1 Manual speed change
  VarChangeFlag%(31) = 0
  I% = CmdServo1%("????????????")
End If

If VarChangeFlag%(32) <> 0 Then 'Inker #2 Auto speed change
  VarChangeFlag%(32) = 0
  I% = CmdServo1%("????????????")
End If

If VarChangeFlag%(33) <> 0 Then 'Inker #2 Manual change
  VarChangeFlag%(33) = 0
  I% = CmdServo1%("????????????")
End If

If VarChangeFlag%(34) <> 0 Then 'Inker #1 mode
  VarChangeFlag%(34) = 0
  I% = CmdServo1%("????????????")
End If

If VarChangeFlag%(35) <> 0 Then 'Inker #2 mode
  VarChangeFlag%(35) = 0
  I% = CmdServo1%("????????????")
End If

If VarChangeFlag%(36) <> 0 Then 'Intaglio Impression mode
  VarChangeFlag%(36) = 0
  I% = CmdServo1%("????????????")
End If

If VarChangeFlag%(37) <> 0 Then 'Intaglio Wiping mode
  VarChangeFlag%(37) = 0
  I% = CmdServo1%("WipeMode=" & Str$(IntaglioWipeControl%))
End If

If VarChangeFlag%(38) <> 0 Then 'Intaglio Prewipe mode
  VarChangeFlag%(38) = 0
  I% = CmdServo1%("????????????")
End If

If VarChangeFlag%(39) <> 0 Then 'Flexo Impression mode
  VarChangeFlag%(39) = 0

```

Module2 - 18

```

    I% = CmdServo1%("????????????")
End If

If VarChangeFlag%(40) <> 0 Then 'Wiping Manual OFF/ON mode
    VarChangeFlag%(40) = 0
    I% = CmdServo1%("WipeMoo=" & Str$(IntaglioWipeManual%))
End If

If VarChangeFlag%(41) <> 0 Then 'Recalculate Cam table
    I% = CmdServo1%("TrigCam=" & Str$(VarChangeFlag%(41)))
    If VarChangeFlag%(41) = 1 Then
        I% = CmdServo2%("DEX=0")
        I% = CmdServo2%("DPX=0")
        I% = CmdServo2%("DPZ=0")
        I% = CmdServo2%("DPW=0")
        I% = CmdServo2%("AxisDXa[0]=0")
        I% = CmdServo2%("AxisDXa[3]=0")
        I% = CmdServo2%("AxisDXb[0]=0")
        I% = CmdServo2%("AxisDXb[3]=0")
        I% = CmdServo2%("AxisX[0]=0")
        I% = CmdServo2%("AxisX[3]=0")
        I% = CmdServo2%("AxisZ[0]=0")
        I% = CmdServo2%("AxisZ[3]=0")
        I% = CmdServo2%("AxisW[0]=0")
        I% = CmdServo2%("AxisW[3]=0")
    End If
    VarChangeFlag%(41) = 0
End If

If VarChangeFlag%(42) <> 0 Then 'Altering IP offset
    VarChangeFlag%(42) = 0
    I% = CmdServo1%("PD " & Trim$(Str$(IP_Move%)))
End If

If VarChangeFlag%(43) <> 0 Then 'Flexo register change
    VarChangeFlag%(43) = 0
    I% = CmdServo2%("Flexoffs=" & Trim$(frmPressrun!MhRealInput4.Text))
End If

If VarChangeFlag%(44) <> 0 Then 'Intaglio Mark window begin
    VarChangeFlag%(44) = 0
    I% = CmdServo1%("IntLwin=" & frmConfig!MhRealInput3.Text)
End If

If VarChangeFlag%(45) <> 0 Then 'Intaglio Mark window end
    VarChangeFlag%(45) = 0
    I% = CmdServo1%("IntUwin=" & frmConfig!MhRealInput10.Text)
End If

If VarChangeFlag%(46) <> 0 Then 'Diecutting Mark window begin
    VarChangeFlag%(46) = 0
    I% = CmdServo1%("DieLwin=" & frmConfig!MhRealInput4.Text)
End If

If VarChangeFlag%(47) <> 0 Then 'Diecutting Mark window end
    VarChangeFlag%(47) = 0
    I% = CmdServo1%("DieUwin=" & frmConfig!MhRealInput12.Text)
End If

If VarChangeFlag%(48) <> 0 Then 'Cutoff Mark window begin
    VarChangeFlag%(48) = 0
    I% = CmdServo2%("CutLwin=" & frmConfig!MhRealInput5.Text)
End If

If VarChangeFlag%(49) <> 0 Then 'Cutoff Mark window end
    VarChangeFlag%(49) = 0
    I% = CmdServo2%("CutUwin=" & frmConfig!MhRealInput1.Text)

```

Module2 - 19

End If

```

If VarChangeFlag%(50) <> 0 Then 'Forces uncalibrate on Product
    VarChangeFlag%(50) = 0
    I% = CmdServoI%("ProdCal=0")
End If

```

End Sub

Public Sub AutoIntegrator()

```

'Routine monitors position error on CF1 axis and makes long term
'gear ratio adjustments to correct print length errors, etc.

```

```

' If CF1error! < 0 Then
'     If LastCF1% < 0 Then
'         CF1errorcnt% = CF1errorcnt% + 1 'This is the # of contiguous errors
'         If CF1errorcnt% > CInt(Param tbl$(928, 2)) Then
'             frmWPsetup!MhRealInput1(5).Text = frmWPsetup!MhRealInput1(5).Text + CDb1(Param_
tbl$(927, 2))
'                 VarChangeFlag%(12) = 1
'                 CF1errorcnt% = 0
'             End If
'         Else
'             LastCF1% = 1
'             CF1errorcnt% = 1
'         End If
'     Else
'         If LastCF1% > 0 Then
'             CF1errorcnt% = CF1errorcnt% + 1
'             If CF1errorcnt% > CInt(Param tbl$(928, 2)) Then
'                 frmWPsetup!MhRealInput1(5).Text = frmWPsetup!MhRealInput1(5).Text - CDb1(Param_
tbl$(927, 2))
'                 VarChangeFlag%(12) = 1
'                 CF1errorcnt% = 0
'             End If
'         Else
'             LastCF1% = -1
'             CF1errorcnt% = 1
'         End If
'     End If
End If

```

End Sub

## Module1 - 1

```

'Program      : General
'Creation date : 02-Apr-97
'Module       : PLC_module.BAS
'Version      : V2.0
'Edit        : -00
'Edit date   : 02-Nov-00
'Author      : Joseph B. Schutte III
              : Industrial Light & Motion, Inc.
              : 2170 Van Blaricum Rd.
              : Cincinnati OH USA
'Copyright (C) 1997-2000 Industrial Light & Motion, Inc.
'Revision History
'
'Ver/Edit   Edit date   Who      Reason
'V2.0-00    02-Nov-00   JBSIII   Rewrite logic for Press_00
'V1.0-01    03-Apr-97   JBSIII   Add Run/Stop function

```

## Option Explicit

PLC Discrete Input Definitions

Parallel I/O channel definitions:

28-Sep-00 11:05

## Inputs

Channel	Function
I00	ESTOP
I01	Intaglio PB: Wiping Jog
I02	Intaglio PB: Press Jog
I03	Intaglio PB: Product Calibrate
I04	Intaglio PB: Wiping Calibrate
I05	Console: STOP
I06	Console: RUN
I07	Console: JOG
I08	Console: Line speed Increase
I09	Console: Line speed Decrease
I10	Console: Press Enable Keyswitch
I11	Console: Product Vacuum Box Blower
I12	Console: Wiping Vacuum Box Blower
I13	Rt Intaglio PB: Press Run
I14	Rt Intaglio PB: Press Jog
I15	Rt Intaglio PB: Press Stop
I16	Console Jam-up stop button
I17	Intaglio Outfeed Webbreak switch
I18	Wired to #17 was Cutoff infeed Webbreak switch
I19	Wiping paper Infeed Webbreak switch
I20	Web Splice detect
I21	Hydraulic Pressure OK sensor
I22	Rt Intaglio PB: Wiping Jog PB
I23	Rt Intaglio PB: Press speed Increase
I24	Rt Intaglio PB: Press speed Decrease
I25	Rt Intaglio PB: Wiping Nip manual OFF/ON
I26	Wiping Drive ready
I27	Main Drive ready
I28	SH1 Overtemp
I29	SH2 Overtemp
I30	Rt Intaglio PB: Prewipe manual Nip Off/On
I31	Rt Intaglio PB: Inker #1 manual Run



## Module1 - 2

```

'I32      Rt Intaglio PB: Inker #2 manual Run
'I33      Rt Intaglio PB: Inker #1 manual Nip Off/On
'I34      Rt Intaglio PB: Inker #2 manual Nip Off/On
'I35      Rt Intaglio PB: Inker #1 manual Run
'I36      Rt Intaglio PB: Inker #2 manual Run
'I37      Rt Intaglio PB: Inker #1 manual Nip Off/On
'I38      Rt Intaglio PB: Inker #2 manual Nip Off/On
'I39      Numbering Unit #1 Off/On Wire #308
-----
'I40      Numbering Unit #2 Off/On Wire #307
'I41      Numbering Unit Web break Wire #306 - Not installed
'I42      Sheeter: Press Jog Wire #303
'I43      Sheeter: Press Stop Wire #304
'I44      Sheeter: Jamup Stop Wire #305
'I45      Intaglio Guard
'I46      Numbering Guard
'I47      Sheeter Guard
-----
'I48
'I49
'I50      Sheeter: Press speed Increase Wire #300
'I51      Sheeter: Press speed Decrease Wire #301
'I52
'I53      Unwind Core sensor
'I54      Delivery Table Down LS
'I55      Sheeter: Press Run Wire #302
-----
'I56      Flexo PB's: Press Stop
'I57      Flexo PB's: Press Jog
'I58      Flexo PB's: Press Run
'I59      Flexo PB's: Increase Line Speed
'I60      Flexo PB's: Decrease Line Speed
'I61      Flexo PB's: Auto/Manual PB
'I62      Flexo PB's: Rewind Forward Select
'I63      Flexo PB's: Rewind Reverse Select
-----
'I64
'I65
'I66      Unwind Web break
'I67      Flexo-Intaglio Web break
'I68      Chill Rolls Web break
'I69      Sheeter Web break
'I70
'I71      Flexo/Die Jam-up stop buttons

```

## PLC Discrete Output Definitions

## Outputs

Channel	Function
'O00	Power ON (Mains)
'O01	Intaglio PB: Product Calibrated
'O02	Intaglio PB: Wiping Calibrated
'O03	Primary Power Transformer Contactor
'O04	Main Drive Contactor
'O05	Main Drive Enable
'O06	Main Drive RAMP Relay
'O07	
'O08	Wired to #35 Web Cleaner / Vac box blower starter
'O09	Inker #2 Drive enable relay
'O10	Main Drive brake relay
'O11	Splice Detector enable
'O12	Hydraulic Pump Contactor
'O13	Hydraulic Pressure LOW mode solenoid

## Module1 - 3

```

'O14      Intaglio Impression ON solenoid
'O15      Inker #1 Drive enable relay
-----
'O16      Inker #1 Nip ON solenoid
'O17      Prewipe Roll engage solenoid
'O18      Wiping Drive Contactor
'O19      Wiping Roll engage solenoid
'O20      Wiping Drive enable
'O21      Inker #2 Nip ON solenoid
'O22      Wiping Rewinder Drive enable
'O23      Wiping Rewinder Brake enable/
-----
'O24      Cutoff Unit Infeed drive enable
'O25      Cutoff Unit Infeed Brake enable/
'O26      Cutoff Unit Drive enable
'O27      Cutoff Unit Brake enable/
'O28      Rt Intaglio PB: Wiping Nip ON
'O29      Rt Intaglio PB: Prewipe Nip ON
'O30      Rt Intaglio PB: Inker #1 Running
'O31      Rt Intaglio PB: Inker #1 Nip ON
-----
'O32      Rt Intaglio PB: Inker #2 Running
'O33      Rt Intaglio PB: Inker #2 Nip ON
'O34      Flexo Impression Solenoid
'O35      Flexo Auto/Manual Solenoid
'O36      Flexo Rewinder Forward Solenoid
'O37      Flexo Rewinder Reverse Solenoid
'O38      Flexo UV Unit Enable solenoid
'O39      Product Vacuum blower solenoid
-----
'O40      Wiping Vacuum blower solenoid
'O41      Scrap blower solenoid
'O42      Numbering Unit 1 ON solenoid
'O43      Numbering Unit 2 ON solenoid
'O44
'O45
'O46      Motion warning bell
'O47

```

```

Global ActualLinSpeed%      'Actual Line Speed in FPM
Global ActualWipingSpeed%   'Actual (commanded) Wiping Speed FPM
Global DecelMode%           'Determines decel rate; normal=0, web break=1
Global FlexoImpsControl%    '0-off 1-manual 2-auto
Global FlexoImpsManual%     '0-off 1-manual on
Global GuardsOK%            'Set when no guard switches open
Global GuardsBypassed%     'Set when operator bypasses switches in Wpsetup
Global Hydraulics%          '0-off 1-low 2-normal
Global IntaglioImpsControl% '0-off 1-manual 2-auto
Global IntaglioImpsManual%  '0-off 1-on
Global IntaglioWipeControl% '0-off 1-manual 2-auto
Global IntaglioWipeManual%  '0-off 1-on
Global IntaglioPrewipeControl% '0-off 1-manual 2-auto
Global IntaglioPrewipeManual% '0-off 1-on
Global IntaglioInker1Control% '0-off 1-manual 2-auto
Global IntaglioInker1nip%   '0-off 1-on
Global IntaglioInker1Manrun% '0-off 1-on
Global IntaglioInker2Control% '0-off 1-manual 2-auto
Global IntaglioInker2nip%   '0-off 1-on
Global IntaglioInker2Manrun% '0-off 1-on
Global JamButtonsOK%        'Intermediate variable set when all jam btms OFF
Global JogRequest%          'Set when operator presses Jog button
Global JogSpeed%            'Press Jog Speed
Global Keyswitch%           'State of console keyswitch
Global LastTlineSpeed%      'Previous scan line TargetLineSpeed%
Global LastIntaglioRmode%   'Last intaglio run mode

```

## Module1 - 4

```

Global LastIntaglioWipemode% 'Last scan intaglio wiping mode
Global LastRunMode% 'Previous scan RunMode%
Global LinSpeedIntegrator% 'Used for speed ramp generator
Global MachineOKtoRun% 'Bit set when machine is ready to run
Global MachineRunning% 'Set when machine in the RUN mode
Global MaxLineSpeed% 'Maximum line speed in strokes per min x 10
Global MinLineSpeed% 'Minimum line speed in strokes per min x 10
Global MSIntaglioJogReq% 'Motion system request for Intaglio Jog
Global MSWipingJogReq% 'Motion system request for Wiping Jog
Global MSWipingImpReq% 'Motion system request for Wiping Imps
Global Numbering_1_on% 'Numbering unit 1 sequencer
Global Numbering_2_on% 'Numbering unit 2 sequencer
Global PilerInhAuto% 'Set when High piler prox switch disabled
Global PLC_RunMode% 'Must be set for PLC code to execute
Global ProductCalRequest% 'Set when Operator presses cal button
Global ProductCalReqS% 'Variable from screen request for cal
Global RequestFastStop% 'Set when a one second stop required
Global RequestNormStop% 'Set when normal stop has been requested
Global RunMode% 'Decoded mode from PLC logic
Global RunRequest% 'Operator has requested run, timing commensing
Global ScrapRequest% 'VB screen activates and we map to Output here
Global SetupOK% 'True when sufficient setup data avail for cal
Global TargetLinSpeed% 'Operator setpoint line speed FPM = Value / 10
Global WebBreaksOK% 'Set when no web break switches tripped
Global WipingCalRequest% 'Set when operator presses cal button
Global WipingCalReqS% 'Variable for signal from screen for req.
Global WipingJogReq% 'Variable for Wiping jog status
Global WipingWinderEna% 'Enable status of wiping rewinder

Dim PLC_t1% 'Temporary working registers
Dim PLC_t2%
Dim PLC_t3%
Dim PLC_t4%
Dim PLC_t5%
Dim PLC_t6%
Dim PLC_t7%
Dim PLC_t8%
Dim PLC_t9%
Dim PLC_t10%

```

## Public Sub PLC\_FirstScan()

'This routine handles all initial settings by being called on powerup.

Dim I%

Call ScanPLCInputs

```

MachineRunning% = 0 'Machine run state; initially stopped
DecelMode% = 0 'Normal stop when 0, Jamup/Webbbreak stop when 1
RequestNormStop% = 0
RequestFastStop% = 0
JogRequest% = 0
WipingCalRequest% = 0
ProductCalRequest% = 0
MachineCalibrated% = 0
FlexoCalibrated% = 0
IntaglioCalibrated% = 0
LastIntaglioRmode% = 0
PilerInhAuto% = 0
JamButtonsOK% = 1
Numbering_1_on% = 0
Numbering_2_on% = 0
GuardsOK% = 1
ScrapRequest% = 0
WipingWinderEna% = 0
frmP00Intaglio!Label57.Visible = True

```

Module1 - 5

```

frmP00Intaglio!Label7.Visible = False
frmP00Intaglio!Label5.Visible = False
Call Hyd_Init_Off
Call IntImps_Init_Off
Call IntWipe_Init_Off
Call IntPrewipe_Init_Off
Call IntInker1_Init_Off
Call IntInker2_Init_Off
WebBreaksOK% = 1
MinLineSpeed% = CInt(Param_tbl$(901, 2))
MaxLineSpeed% = CInt(Param_tbl$(902, 2))
frmPressrun!Slider1.Value = MinLineSpeed% 'Get initial speed from parameter table
PLC_O%(0) = 0 'Main Power Off
PLC_O%(1) = 0 'Kill Product calibrated light
PLC_O%(2) = 0 'Kill Wiping calibrated light
PLC_O%(3) = 0 'Turn off primary xformer
PLC_O%(4) = 0 'Turn off Main drive contactor
PLC_O%(5) = 0 'Turn off main drive enable
PLC_O%(6) = 0 'Shift main drive ramp to fast
PLC_O%(7) = 0 'Not Used
PLC_O%(8) = 0 'Scrap Blower primary
PLC_O%(9) = 0 'Disable #2 Inker
PLC_O%(10) = 0 'Engage Main drive brake
PLC_O%(11) = 0 'Disable Splice Detector
PLC_O%(12) = 0 'Turn off hydraulic pump
PLC_O%(13) = 1 'Shift hydraulic pressure to low
PLC_O%(14) = 0 'Turn off Intaglio impressions
PLC_O%(15) = 0 'Disable Inker #1 drive
PLC_O%(16) = 0 'Turn OFF Inker #1 Nip
PLC_O%(17) = 0 'Disengage Prewipe roller
PLC_O%(18) = 0 'Turn off Wiping drive contactor
PLC_O%(19) = 0 'Disengage Wiping roller
PLC_O%(20) = 0 'Disable Wiping drive
PLC_O%(21) = 0 'Not Used
PLC_O%(22) = 0 'Disable Wiping rewind drive
PLC_O%(23) = 0 'Enable Wiping rewind brake
PLC_O%(24) = 0 'Disable Cutoff Unit infeed drive
PLC_O%(25) = 0 'Enable Cutoff unit infeed brake
PLC_O%(26) = 0 'Disable Cutoff unit drive
PLC_O%(27) = 0 'Enable Cutoff Unit brake
PLC_O%(28) = 0 'Kill Wiping nip on light
PLC_O%(29) = 0 'Kill Prewipe nip on light
PLC_O%(30) = 0 'Kill Inker #1 running light
PLC_O%(31) = 0 'Kill Inker #1 nip on light
PLC_O%(32) = 0 'Kill Inker #2 running light
PLC_O%(33) = 0 'Kill Inker #2 nip on light
PLC_O%(34) = 0 'Turn off Flexo Impression soleniod
PLC_O%(35) = 0 'Select Flexo Manual mode
PLC_O%(36) = 0 'Turn off Flexo Rewinder forward solenoid
PLC_O%(37) = 0 'Turn off Flexo Rewinder reverse solenoid
PLC_O%(38) = 0 'Disable Flexo UV System
PLC_O%(39) = 0 'Turn off Product vacuum blower
PLC_O%(40) = 0 'Turn off Wiping vacuum blower
PLC_O%(41) = 0 'Turn off scrap blower
PLC_O%(42) = 0 'Not used
PLC_O%(43) = 0 'Not used
PLC_O%(44) = 0 'Not used
PLC_O%(45) = 0 'Not used
PLC_O%(46) = 0 'Motion warning bell
PLC_O%(47) = 0 'Not used

```

```

Call StrobePLCOutputs

```

```

PLC_RunMode% = 1 'Enable scanning

```

```

End Sub

```

Module1 - 6

Public Sub PLC\_Scan()

'This routine gets called every N milliseconds by the MAIN.FRM PLC timer. The equations will be evaluated only if RunMode% is set as 'non-zero.

Dim I%, J%

MachineCalibrated% = 255 '????????????? Temporary force OK

If PLC\_RunMode% &lt;&gt; 0 Then

Call ScanPLCInputs 'Read the PLC inputs

If PLC\_I%(0) = 0 Then

Call PLC\_FirstScan

Else 'If we're ESTOP do nothing but hold off outputs

If (PLC\_I%(0) &lt;&gt; 0) And (PLC\_I%(1) = 0) And (PLC\_I%(2) = 0) And (PLC\_O%(0) = 0) Then

PLC\_O%(0) = 1 'Systemwide power on flag

PLC\_O%(3) = 1 'Primary transformer contactor

End If

'Look for Main power on request

If SysMode% = 0 Then 'Flexo only mode

PLC\_O%(4) = 0 'Main drive contactor

PLC\_O%(18) = 0 'Wiping drive contactor

Else 'Combi mode - turn on drives

PLC\_O%(4) = 1

PLC\_O%(18) = 1

End If

If (((PLC\_I%(44) &lt;&gt; 0) Or (PLC\_I%(16) = 0) Or (PLC\_I%(71) = 0)) And JamButtonsOK% = 1) Then

JamButtonsOK% = 0

RequestFastStop% = 1

End If

'Handle Jam up Stop buttons / switches

If ((PLC\_I%(44) = 0) And JamButtonsOK% = 0) Then

JamButtonsOK% = 1

End If

'Detect end of jam up stop condition

If ((PLC\_I%(6) &lt;&gt; 0) Or (PLC\_I%(13) &lt;&gt; 0) Or (PLC\_I%(55) &lt;&gt; 0) Or (PLC\_I%(58) &lt;&gt; 0)) And (MachineOKtoRun% &lt;&gt; 0) And (MachineRunning% = 0) Then

RunRequest% = 1

Else

RunRequest% = 0

End If

'Handle Run buttons

If (PLC\_I%(5) &lt;&gt; 0) Or (PLC\_I%(15) &lt;&gt; 0) Or (PLC\_I%(43) &lt;&gt; 0) Or (PLC\_I%(56) = 0) Then

RunRequest% = 0 'In case they just pressed the Run button

RequestNormStop% = 1 'Sequence machine to stop

End If

'Handle the Normal Stop buttons

'If (PLC\_I%(21) &lt;&gt; 0) Or (PLC\_I%(54) &lt;&gt; 0) Then

'RunRequest% = 0 'In case they just pressed the Run button

'RequestNormStop% = 1 'Sequence machine to stop

'End If

'Handle the Machine induced stops that do not require Fast Stops

If ((PLC\_I%(2) &lt;&gt; 0) Or (PLC\_I%(7) &lt;&gt; 0) Or (PLC\_I%(14) &lt;&gt; 0) Or (PLC\_I%(42) &lt;&gt; 0) Or (PLC\_I%(57) &lt;&gt; 0)) And (MachineRunning% = 0) And (JamButtonsOK% &lt;&gt; 0) Then

Module1 - 7

```

    JogRequest% = 1
End If
'Handle the Jog buttons Machine must be not be running or in jam up stop state
'Jog and Increase buttons together cause Turbo jog 4X speed

If ((PLC_I%(2) = 0) And (PLC_I%(7) = 0) And (PLC_I%(14) = 0) And (PLC_I%(42) = 0) And (P
LC_I%(57) = 0) And (JogRequest% = 1)) Then
    JogRequest% = 0 'Terminate Jog if they released the button
End If

If ((PLC_I%(1) <> 0) Or (PLC_I%(22) <> 0)) And (IntaglioWipeControl% = 1) And (MachineRu
nning% = 0) Then 'Wiping Jog PB pressed
    WipingJogReq% = 1
End If
'Look for a Wiping Jog button

If (PLC_I%(1) = 0) And (PLC_I%(22) = 0) And (WipingJogReq% = 1) Then
    WipingJogReq% = 0
End If
'Look for Wiping Jog button release

If ((PLC_I%(3) <> 0) Or (ProductCalReqS% <> 0)) And (MachineRunning% = 0) And (JogReques
t% = 0) Then
    ProductCalRequest% = 1 'Set the calibrate status bit
End If
'Detect Operator Cal product request

If ((PLC_I%(3) = 0) And (ProductCalReqS% = 0)) Or (ProductCalibrated% = 255) Then
    ProductCalRequest% = 0 'Clear the Homing request
End If
'Operator has released the Calibrate / Homing button

If ((PLC_I%(4) <> 0) Or (WipingCalReqS% <> 0)) And (MachineRunning% = 0) And (JogRequest
% = 0) Then
    WipingCalRequest% = 1 'Set the calibrate status bit
End If
'Detect Operator Cal wiping request

If ((PLC_I%(4) = 0) And (WipingCalReqS% = 0)) Or (WipingCalibrated% = 255) Then
    WipingCalRequest% = 0 'Clear the calibrate request
End If
'Operator has released the Calibrate / Homing button

If ProductCalibrated% = 255 Then
    PLC_O%(1) = 1 'Activate Product calibrated light
    frmP00Intaglio!Label1.Visible = True
    frmPressrun!Label39.BackColor = &HFF00& 'Green
Else
    PLC_O%(1) = 0
    frmP00Intaglio!Label1.Visible = False
    frmPressrun!Label39.BackColor = &HFFFFFFF
End If
'Monitor product calibrated status and map to light

If WipingCalibrated% = 255 Then
    PLC_O%(2) = 1 'Activate Wiping calibrated light
    frmP00Intaglio!Label24.Visible = True
    frmPressrun!Label40.BackColor = &HFF00& 'Green
Else
    PLC_O%(2) = 0
    frmP00Intaglio!Label24.Visible = False
    frmPressrun!Label40.BackColor = &HFFFFFFF 'White
End If
'Monitor Wiping calibrated status and map to light

J% = PLC_I%(17) + PLC_I%(66) + PLC_I%(67) + PLC_I%(68)
If CutoffEnable% <> 0 Then J% = J% + PLC_I%(69) 'Use sheeter if enabled

```

Module1 - 8

```

If NumberingEnable% <> 0 Then J% = J% + PLC_I%(41) 'Use numbering if enabled
If (J% <> 0) Or ((PLC_I%(19) <> 0) And (SysMode% <> 1)) Then
    WebBreaksOK% = 0
Else
    WebBreaksOK% = 1
End If
'Monitor Web break switches; 1 = OK    0 = Break
'Only test wiping if we're using it

If ((PLC_I%(45) = 0) Or (PLC_I%(46) = 0) Or (PLC_I%(47) = 0)) Then
    GuardsOK% = 0
Else
    GuardsOK% = 1
End If
If GuardsBypassed% = 1 Then GuardsOK% = 1 'Operator bypass on P00Flexo
'Monitor guard switches; 1 = OK    0 = Door open

If ((WebBreaksOK% = 0) Or (GuardsOK% = 0) Or (JamButtonsOK% = 0) Or (MachineCalibrated%
<> 255)) Then
    RunRequest% = 0
    MachineOKtoRun% = 0 'Some fault condition exists
Else
    MachineOKtoRun% = 1 'All subsystems OK
End If
'Monitor all machine fault detectors

If (MachineRunning% = 2) And (MachineOKtoRun% = 0) And (RequestFastStop% = 0) Then
    RequestFastStop% = 1
End If
'Monitor Web break and guard switches

Keyswitch% = PLC_I%(10) 'Map Keyswitch to variable
If Keyswitch% <> 0 Then
    frmPressrun!Label20.Visible = True
Else
    frmPressrun!Label20.Visible = False
End If

If ((PLC_I%(8) <> 0) Or (PLC_I%(23) <> 0) Or (PLC_I%(50) <> 0) Or (PLC_I%(59) <> 0)) The
n
    If ((PLC_I%(2) <> 0) Or (PLC_I%(7) <> 0) Or (PLC_I%(14) <> 0) Or (PLC_I%(42) <> 0) O
r (PLC_I%(57) <> 0)) Then
        TargetLinSpeed% = TargetLinSpeed% + 0
    Else
        I% = TargetLinSpeed% + 1
        frmPressrun!Slider1.Value = I%
    End If
End If
'Handle Increase speed buttons. 4000 = 400 Feet per min

If ((PLC_I%(9) <> 0) Or (PLC_I%(24) <> 0) Or (PLC_I%(51) <> 0) Or (PLC_I%(60) <> 0)) The
n
    I% = TargetLinSpeed% - 1
    frmPressrun!Slider1.Value = I%
End If
'Handle Decrease speed button

If ((PLC_I%(8) <> 0) And (PLC_I%(9) <> 0)) Or ((PLC_I%(23) <> 0) And (PLC_I%(24) <> 0))
Then
    frmPressrun!Slider1.Value = 25
End If
If ((PLC_I%(50) <> 0) And (PLC_I%(51) <> 0)) Or ((PLC_I%(59) <> 0) And (PLC_I%(60) <> 0)
) Then
    frmPressrun!Slider1.Value = 25
End If
'Hidden speed jump to 25 FPM

```

Module1 - 9

```

    If ((PLC_I%(20) <> 0) And (PLC_O%(11) <> 0) And (MachineRunning% <> 0) And (MachineRunni
ng% <> 5)) Then
        RequestFastStop% = 1
    End If
    'Detect splices when machine is running

    If ((PLC_I%(53) <> 0) And (MachineRunning% <> 0) And (MachineRunning% <> 5)) Then
        RequestFastStop% = 1
    End If
    'Mill roll out (core sensor)

    If ((PLC_I%(62) <> 0) And (PLC_I%(63) = 0)) Then
        PLC_O%(36) = 1
    Else
        PLC_O%(36) = 0
    End If
    'Control product rewinder forward solenoid

    If ((PLC_I%(63) <> 0) And (PLC_I%(62) = 0)) Then
        PLC_O%(37) = 1
    Else
        PLC_O%(37) = 0
    End If
    'Control product rewinder reverse solenoid

    If (WipingWinderEna% = 0) And (PLC_O%(22) <> 0) Then
        PLC_O%(22) = 0 'Kill enable
        PLC_O%(23) = 0 'Enable brake
    End If
    If (WipingWinderEna% > 0) And (PLC_O%(22) = 0) Then
        PLC_O%(22) = 1 'Kill enable
        PLC_O%(23) = 1 'Enable brake
    End If
    ' This the Wiping rewinder logic

    PLC_O%(39) = PLC_I%(11) 'Map the Product Vacuum blower
    PLC_O%(40) = PLC_I%(12) 'Map the Wiping Vacuum blower

    If ((PLC_I%(39) <> 0) And (PLC_I_Last%(39) = 0)) Then
        Call Numbers1_Man_Toggle
    End If
    If ((PLC_I%(40) <> 0) And (PLC_I_Last%(40) = 0)) Then
        Call Numbers2_Man_Toggle
    End If
    PLC_O%(42) = Numbering_1_on%
    PLC_O%(43) = Numbering_2_on%
    'Map the Numbering unit controls

    PLC_O%(41) = ScrapRequest%
    PLC_O%(8) = ScrapRequest%

    Select Case Hydraulics% 'Hydraulic System Control
        Case 0 'Off
            PLC_O%(12) = 0 'Pump motor
            PLC_O%(13) = 0 'Pressure control
        Case 1 'Low pressure
            PLC_O%(12) = 1
            PLC_O%(13) = 1
        Case 2 'Normal pressure
            PLC_O%(12) = 1
            PLC_O%(13) = 0
    End Select
    '-----

    Select Case FlexoImpsControl% 'Flexo Impression Control
        Case 0 ' Off
            PLC_O%(35) = 0 'Manual control

```



Module1 - 10

```

    PLC_O%(34) = 0 'Turn off output
    Case 1 ' Manual
    PLC_O%(35) = 1 ' Auto solenoid
    If ((PLC_I%(61) <> 0) And (PLC_I_Last%(61) = 0)) Then
        Call FlexoImps_Man_Toggle
    End If
    PLC_O%(34) = FlexoImpsManual%
    Case 2 'Auto
    PLC_O%(35) = 1 'Auto solenoid on
    If (ActualLinSpeed% > Param_tbl$(909, 2)) Then
        PLC_O%(34) = 1
    Else
        PLC_O%(34) = 0
    End If
End Select
'-----

Select Case IntaglioImpsControl% 'Intaglio Impression Control
    Case 0 ' Off
    PLC_O%(14) = 0 'Turn off output
    Case 1 ' Manual
    PLC_O%(14) = IntaglioImpsManual%

    Case 2 'Auto
    If (ActualLinSpeed% > Param_tbl$(910, 2)) Then
        PLC_O%(14) = 1
    Else
        PLC_O%(14) = 0
    End If
End Select
'-----

Select Case IntaglioWipeControl% 'Intaglio Wiping Control

    Case 0 ' Off
    PLC_O%(19) = 0 'Turn off output
    PLC_O%(28) = 0 'Kill the light

    Case 1 ' Manual
    If ((PLC_I%(25) <> 0) And (PLC_I_Last%(25) = 0)) Then
        Call IntWipe_Man_Toggle
    End If
    PLC_O%(19) = IntaglioWipeManual%
    PLC_O%(28) = IntaglioWipeManual%
    If (WipingCalRequest% <> 0) And (MSWipingImpReq% <> 0) Then
        PLC_O%(19) = 1 'Force if calibrating
        PLC_O%(28) = 1
    End If

    Case 2 'Auto
    If MSWipingImpReq% <> 0 Then
        PLC_O%(19) = 1
        PLC_O%(28) = 1
    Else
        PLC_O%(19) = 0
        PLC_O%(28) = 0
    End If
End Select
'-----

Select Case IntaglioPrewipeControl% 'Intaglio Prewipe Control
    Case 0 ' Off
    PLC_O%(17) = 0 'Turn off output
    PLC_O%(29) = 0 'Kill the light
    Case 1 ' Manual
    If ((PLC_I%(30) <> 0) And (PLC_I_Last%(30) = 0)) Then

```

Module1 - 11

```

        Call IntPrewipe_Man_Toggle
    End If
    PLC_O%(17) = IntaglioPrewipeManual%
    PLC_O%(29) = IntaglioPrewipeManual%
    Case 2 'Auto
    If (ActualLinSpeed% > Param_tbl$(912, 2)) Then
        PLC_O%(17) = 1
        PLC_O%(29) = 1
    Else
        PLC_O%(17) = 0
        PLC_O%(29) = 0
    End If
End Select
'-----

Select Case IntaglioInker1Control% 'Intaglio Inker #1 Control
    Case 0 ' Off
        PLC_O%(15) = 0 'Turn off drive enable
        PLC_O%(16) = 0 'Turn off nip output
        PLC_O%(30) = 0 'Turn off the lights
        PLC_O%(31) = 0
    Case 1 ' Manual
        If ((PLC_I%(33) <> 0) And (PLC_I_Last%(33) = 0)) Then
            Call Inker1nip_Man_Toggle
        End If
        PLC_O%(16) = IntaglioInker1nip%
        PLC_O%(31) = IntaglioInker1nip%
        If ((PLC_I%(31) <> 0) And (PLC_I_Last%(31) = 0)) Then
            Call Inker1run_Man_Toggle
        End If
        PLC_O%(15) = IntaglioInker1Manrun%
        PLC_O%(30) = IntaglioInker1Manrun%
    Case 2 'Auto
        PLC_O%(15) = 1
        PLC_O%(30) = 1
        'Enable on run - speed calc'ed elsewhere
        If (ActualLinSpeed% > Param_tbl$(915, 2)) Then
            PLC_O%(16) = 1
            PLC_O%(31) = 1
        Else
            PLC_O%(16) = 0
            PLC_O%(31) = 0
        End If
    End Select
'-----

Select Case IntaglioInker2Control% 'Intaglio Inker #2 Control
    Case 0 ' Off
        PLC_O%(9) = 0 'Turn off drive enable
        PLC_O%(21) = 0 'Turn off nip output
        PLC_O%(32) = 0 'Turn off the lights
        PLC_O%(33) = 0
    Case 1 ' Manual
        If ((PLC_I%(34) <> 0) And (PLC_I_Last%(34) = 0)) Then
            Call Inker2nip_Man_Toggle
        End If
        PLC_O%(21) = IntaglioInker2nip%
        PLC_O%(33) = IntaglioInker2nip%
        If ((PLC_I%(32) <> 0) And (PLC_I_Last%(32) = 0)) Then
            Call Inker2run_Man_Toggle
        End If
        PLC_O%(9) = IntaglioInker2Manrun%
        PLC_O%(32) = IntaglioInker2Manrun%
    Case 2 'Auto
        PLC_O%(9) = 1
        PLC_O%(32) = 1
        'Enable on run - speed calc'ed elsewhere

```

Module1 - 12

```

        If (ActualLinSpeed% > Param_tbl$(915, 2)) Then
            PLC_O$(21) = 1
            PLC_O$(33) = 1
        Else
            PLC_O$(21) = 0
            PLC_O$(33) = 0
        End If
    End Select
    '-----

    End If 'End of the ESTOP else if thingy

    Call StrobePLCOutputs 'Update the PLC outputs

End If

End Sub

Public Sub UpdateIntaglioDrive()

    ' Routine writes I/O needed control Intaglio Main drive

    If SysMode% <> 0 Then 'We're in combi mode
        Select Case RunMode%

            Case 0 'Normal Stop mode
                If LastIntaglioRmode% <> 0 Then 'We're initiating
                    LastIntaglioRmode% = 0
                End If
                If LinSpeedIntegrator! > 0 Then
                    LinSpeedIntegrator! = LinSpeedIntegrator! - Val(Param_tbl$(905, 2))
                End If
                Call AioOut(0, LinSpeedIntegrator! * Param_tbl$(923, 2) * (24 / frmConfig!MhRealInput7.V
valueReal))
                If ((ActualLinSpeed% < 3) And (PLC_O$(10) <> 0)) Then
                    PLC_O$(5) = 0 'Disable drive
                    PLC_O$(10) = 0 'Enable brake
                End If

            Case 1 'Calibration mode
                If LastIntaglioRmode% <> 1 Then 'We're initiating
                    LastIntaglioRmode% = 1
                End If
                If IntaglioCalReq% <> 0 Then 'We need to move Main
                    PLC_O$(5) = 1 'Enable drive
                    PLC_O$(6) = 0 'Fast ramp
                    PLC_O$(10) = 1 'Brake off
                    Call AioOut(0, Val(Param_tbl$(908, 2)) * Param_tbl$(923, 2))
                Else
                    PLC_O$(5) = 0 'Enable drive
                    PLC_O$(10) = 0 'Brake off
                    Call AioOut(0, 0)
                End If

            Case 2 'Normal Run mode
                If LastIntaglioRmode% <> 2 Then 'We're initiating
                    PLC_O$(5) = 1 'Enable drive
                    PLC_O$(6) = 1 'Slow ramp
                    PLC_O$(10) = 1 'Brake off
                    LastIntaglioRmode% = 2
                End If
                If LinSpeedIntegrator! < TargetLinSpeed% Then
                    LinSpeedIntegrator! = LinSpeedIntegrator! + Val(Param_tbl$(905, 2))
                End If
                If LinSpeedIntegrator! > TargetLinSpeed% Then

```

Module1 - 13

```

        LinSpeedIntegrator! = LinSpeedIntegrator! - Val(Param_tbl$(905, 2))
    End If
    Call AioOut(0, LinSpeedIntegrator! * Param_tbl$(923, 2) * (24 / frmConfig!MhRealInput7.V
alueReal))

    Case 3 'Fast Stop
        If LastIntaglioRmode% <> 3 Then 'We're initiating
            Call AioOut(0, 0) 'Null the command
            PLC_O%(6) = 0 'Fast ramp
            PLC_O%(5) = 0 'Disable drive
            PLC_O%(10) = 0 'Apply full brake
            LinSpeedIntegrator! = 0
            LastIntaglioRmode% = 3
        End If

    Case 5 ' Jog mode
        If LastIntaglioRmode% <> 5 Then 'Init
            PLC_O%(5) = 1 'Enable drive
            PLC_O%(6) = 1 'Slow ramp
            PLC_O%(10) = 1 'Brake off
            LastIntaglioRmode% = 5
        End If
        If ((PLC_I%(8) <> 0) Or (PLC_I%(23) <> 0) Or (PLC_I%(50) <> 0) Or (PLC_I%(59) <> 0)) The
n
            Call AioOut(0, JogSpeed% * 4 * Param_tbl$(923, 2))
        Else
            Call AioOut(0, JogSpeed% * Param_tbl$(923, 2))
        End If
    End Select

End If

End Sub

Public Sub UpdateWipingDrive()

    'Routine handles logic for Intaglio Wiping Drive

    Dim I%

    If (SysMode% <> 0) Then
        Select Case IntaglioWipeControl% 'Decode the mode

        Case 0 'Offline mode
            If LastIntaglioWipemode% <> 0 Then
                Call AioOut(1, 0) 'Command zero speed
                PLC_O%(20) = 0 'Disable Drive
                PLC_O%(18) = 0 ' Turn off contactor
                LastIntaglioWipemode% = 0
            End If

        Case 1 'Manual mode
            If LastIntaglioWipemode% <> 1 Then
                PLC_O%(18) = 1 'Fire up drive
                LastIntaglioWipemode% = 1
            End If
            If (RunMode% = 0) And (MSWipingJogReq% = 0) And (ActualLinSpeed% = 0) And (PLC_O%(20) <>
0) Then 'We're stopped waiting for jog or ...
                ActualWipingSpeed% = 0
                PLC_O%(20) = 0 'Disable drive
                Call AioOut(1, 0) 'Zero velocity command
            End If
            If ((RunMode% = 0) And (ActualLinSpeed% > 0)) Or (RunMode% = 2) Then 'Generate command
                If PLC_O%(20) = 0 Then PLC_O%(20) = 1 'Be sure drive enabled
                ActualWipingSpeed% = ActualLinSpeed% * frmP00Intaglio!Slider5.Value / 100
                ActualWipingSpeed% = ActualWipingSpeed% * Val(Param_tbl$(832, 2)) / System_circumfer
ence#
            End If
        End Select
    End If

```

Module1 - 14

```

        I% = ActualWipingSpeed% * Val(Param_tbl$(924, 2))
        Call AioOut(1, I%)
    End If
    If (RunMode% = 8) And (MSWipingJogReq% <> 0) Then 'We're jogging
        PLC_O%(20) = 1 'Enable drive
        ActualWipingSpeed% = Val(Param_tbl$(925, 2))
        I% = Val(Param_tbl$(924, 2)) * Val(Param_tbl$(925, 2))
        'Wiping velocity scale x Wiping jog speed
        Call AioOut(1, I%) 'Jog velocity command
    End If
    If (RunMode = 9) And (MSWipingJogReq% <> 0) Then 'We're calibrating
        PLC_O%(20) = 1 'Enable drive
        ActualWipingSpeed% = Val(Param_tbl$(929, 2))
        I% = Val(Param_tbl$(924, 2)) * Val(Param_tbl$(929, 2))
        'Wiping velocity scale x Wiping calibrate speed
        Call AioOut(1, I%) 'Jog velocity command
    End If

Case 2 'Auto mode
    If LastIntaglioWipemode% <> 2 Then
        PLC_O%(18) = 1 'Fire up drive
        LastIntaglioWipemode% = 2
    End If
    If ((RunMode% = 0) Or (RunMode% = 3)) And (ActualLinSpeed% > 0) Or (RunMode% = 2) Then
'Generate command
        If PLC_O%(20) = 0 Then PLC_O%(20) = 1 'Be sure drive enabled
        ActualWipingSpeed% = ActualLinSpeed% * frmP00IntaglioSlider5.Value / 100
        ActualWipingSpeed% = ActualWipingSpeed% * Val(Param_tbl$(832, 2)) / System_circumfer
ence#
        I% = ActualWipingSpeed% * Val(Param_tbl$(924, 2))
        Call AioOut(1, I%)
    End If
    If (RunMode% = 9) And (MSWipingJogReq% <> 0) Then 'Motion sys req
        If PLC_O%(20) = 0 Then PLC_O%(20) = 1 'Be sure drive enabled
        ActualWipingSpeed% = Val(Param_tbl$(929, 2))
        I% = ActualWipingSpeed% * Val(Param_tbl$(924, 2))
        Call AioOut(1, I%)
    End If
    If ((RunMode% = 0) Or (RunMode% = 3)) And (ActualLinSpeed% = 0) And (MSWipingJogReq% = 0
) Then 'Stopped
        If PLC_O%(20) <> 0 Then PLC_O%(20) = 0 'disable wipoing drive
        ActualWipingSpeed% = 0
    End If

End Select

Else 'We're in Flexo only mode
    PLC_O%(20) = 0 'Disable Drive
    PLC_O%(18) = 0 ' Turn off contactor
    LastIntaglioWipemode% = 0
End If

End Sub

Public Sub UpdateMotionSystemRL()

    'This subroutine evaluates the PLC data and issues appropriate
    'commands to the motion cards

    Dim C%, I%, S%, X, X1

    If (EnableHostPolling1% = True) And (EnableHostPolling2% = True) Then

        S% = RequestNormStop% + RequestFastStop% + Keyswitch% 'Combine variables

        C% = WipingCalibrated%
        If IntaglioWipeControl% <> 2 Then C% = 255

```

Module1 - 15

```

'Mask calibration status if it's not in Auto

If (SystemStatus% And &H300) = &H100 Then
  SystemStatus% = SystemStatus% Or &H200
  I% = CmdServo2%("SysRun=0")
  'We're powering up
End If

If (SystemStatus% And &H300) = &H200 Then
  SystemStatus% = SystemStatus% And &HFCFF
  I% = CmdServo2%("SysRun=9")
  ' We're powering down
End If

If (JogRequest% <> 0) And ((RunMode% = 0) Or (RunMode% = 3)) And (S% = 0) And (Startdelay% = 0) Then
  Startdelay% = 1
  frmMain!Timer1.Enabled = True
  PLC_O%(46) = 1 'start the bell
End If
'Arm the motion timer
If (JogRequest% = 0) And (RunRequest% = 0) And (Startdelay% <> 0) Then
  frmMain!Timer1.Enabled = False
  Startdelay% = 0
  PLC_O%(46) = 0
End If
'Buttons released - Disarm the motion timer

If (JogRequest% <> 0) And ((Startdelay% = 2) Or (Ridethrudelay% <> 0)) Then
  frmMain!Timer1.Enabled = False 'disable timer if not already
  frmMain!Timer6.Enabled = False 'disable ride thru too
  Startdelay% = 0
  Ridethrudelay% = 0
  PLC_O%(46) = 0
  I% = CmdServo1%("SysStat=0") 'Cancel calibration status
  I% = CmdServo1%("SysRun=5")
  I% = CmdServo2%("SysRun=5")
  LastRunMode% = RunMode%
  RunMode% = 5
  MachineRunning% = 5
  RunRequest% = 0
End If
'Jog button pressed

If (JogRequest% = 0) And (RunMode% = 5) Then
  I% = CmdServo1%("SysRun=0")
  I% = CmdServo2%("SysRun=0")
  LastRunMode% = RunMode%
  RunMode% = 0
  MachineRunning% = 0
  Ridethrudelay% = 1 'Arm ride thru timer
  frmMain!Timer6.Enabled = True
End If
'Jog button released

If (RunRequest% = 1) And (RunMode% = 0) And (ProductCalibrated% <> 0) And (S% = 0) And (C% <> 0) And (Startdelay% = 0) Then
  Startdelay% = 1
  frmMain!Timer1.Enabled = True
  PLC_O%(46) = 1
End If
If (RunRequest% = 1) And ((Startdelay% = 2) Or (Ridethrudelay% <> 0)) Then
  frmMain!Timer1.Enabled = False
  frmMain!Timer6.Enabled = False
  Startdelay% = 0
  Ridethrudelay% = 0
  PLC_O%(46) = 0

```

Module1 - 16

```

    I% = CmdServo1$("SysRun=2")
    I% = CmdServo2$("SysRun=2")
    RunMode% = 2
    MachineRunning% = 2
    RunRequest% = 0
End If
'Run button pressed

If (RequestFastStop% <> 0) And (ActualLinSpeed% > 5) Then
    I% = CmdServo1$("SysRun=0")
    I% = CmdServo2$("SysRun=0")
    LastRunMode% = RunMode%
    RunMode% = 3
    MachineRunning% = 0
End If
RequestFastStop% = 0
'Either Jamup PB or Web break occurred

If (RequestNormStop% <> 0) And (MachineRunning% <> 0) Then
    I% = CmdServo1$("SysRun=0")
    I% = CmdServo2$("SysRun=0")
    LastRunMode% = RunMode%
    RunMode% = 0
    MachineRunning% = 0
End If
RequestNormStop% = 0
'Normal Stop button pressed

If (WipingJogReq% <> 0) And (MachineRunning% = 0) Then
    I% = CmdServo1$("SysRun=8")
    I% = CmdServo2$("SysRun=8")
    LastRunMode% = RunMode%
    RunMode% = 8
    MachineRunning% = 8
End If
'Wiping Jog button pressed

If (WipingJogReq% = 0) And (RunMode% = 8) Then
    I% = CmdServo1$("SysRun=0")
    I% = CmdServo2$("SysRun=0")
    LastRunMode% = RunMode%
    RunMode% = 0
    MachineRunning% = 0
End If
'Wiping Jog button released

If WipingWinderEna% = 2 Then 'We're in AUTO
    X = 1 - (PWRewindDia - Param_tbl$(962, 2)) / (Param_tbl$(963, 2) - Param_tbl$(962, 2))
    X1 = (frmConfig!MhRealInput16.ValueReal - frmConfig!MhRealInput15.ValueReal) * X
    X1 = X1 + frmConfig!MhRealInput15.ValueReal
    frmP00Intaglio!Slider6.Value = X1
End If
Call AioOut(2, frmP00Intaglio!Slider6.Value * Param_tbl$(926, 2))
'Update the Intaglio Wiping rewinder vel command

If (ProductCalRequest% <> 0) And (RunMode% = 0) And (S% = 0) Then
    I% = CmdServo1$("SysRun=1")
    I% = CmdServo2$("SysRun=1")
    LastRunMode% = RunMode%
    RunMode% = 1
    MachineRunning% = 1
End If
'Operator has pressed the Product Calibrate button

If (RunMode% = 1) And ((ProductCalRequest% = 0) Or (ProductCalibrated% = 255)) Then
    I% = CmdServo1$("SysRun=0")

```

Module1 - 17

```

        I% = CmdServo2%("SysRun=0")
        LastRunMode% = RunMode%
        RunMode% = 0
        MachineRunning% = 0
    End If
    'Operator released Product cal button or we're calibrated

    If (WipingCalRequest% <> 0) And (RunMode% = 0) And (S% = 0) Then
        I% = CmdServo1%("SysRun=9")
        I% = CmdServo2%("SysRun=9")
        LastRunMode% = RunMode%
        RunMode% = 9
    End If
    'Operator has pressed the Wiping Calibrate button

    If (RunMode% = 9) And ((WipingCalRequest% = 0) Or (WipingCalibrated% = 255)) Then
        I% = CmdServo1%("SysRun=0")
        I% = CmdServo2%("SysRun=0")
        LastRunMode% = RunMode%
        RunMode% = 0
        MachineRunning% = 0
    End If
    'Operator released Wiping cal button or we're calibrated
End If

End Sub
Public Sub Hyd_Init_Off()

    'Turn hydraulics off
    frmP00Intaglio!Label51.Visible = True
    frmP00Intaglio!Label52.Visible = False
    frmP00Intaglio!Label53.Visible = False
    Hydraulics% = 0

End Sub

Public Sub IntImps_Init_Off()

    'Intaglio Impression mode: Off
    If MachineRunning% = 0 Then
        IntaglioImpsControl% = 0
        frmP00Intaglio!Label9.Visible = True
        frmP00Intaglio!Label8.Visible = False
        frmP00Intaglio!Label6.Visible = False
        VarChangeFlag%(36) = 1
        If IntaglioImpsManual% <> 0 Then
            IntaglioImpsManual% = 0
            frmP00Intaglio!Label19.Visible = False
        End If
        frmP00Intaglio!Command22.Enabled = False
    End If

End Sub

Public Sub IntWipe_Init_Off()

    'Intaglio Wiping mode: Off
    If MachineRunning% = 0 Then
        IntaglioWipeControl% = 0
        frmP00Intaglio!Label15.Visible = True
        frmP00Intaglio!Label14.Visible = False
        frmP00Intaglio!Label13.Visible = False
        VarChangeFlag%(37) = 1
        If IntaglioWipeManual% <> 0 Then
            IntaglioWipeManual% = 0
            frmP00Intaglio!Label16.Visible = False
            frmP00Intaglio!Command25.Enabled = False
        End If
    End If

```



Module1 - 18

End If

End Sub

Public Sub IntPrewipe\_Init\_Off()

'Intaglio Prewipe mode: Off

If MachineRunning% = 0 Then

IntaglioPrewipeControl% = 0

frmP00Intaglio!Label21.Visible = True

frmP00Intaglio!Label20.Visible = False

frmP00Intaglio!Label18.Visible = False

VarChangeFlag%(38) = 1

If IntaglioPrewipeManual% &lt;&gt; 0 Then

IntaglioPrewipeManual% = 0

frmP00Intaglio!Label22.Visible = False

End If

frmP00Intaglio!Command29.Enabled = False

End If

End Sub

Public Sub IntImps\_Man\_Toggle()

'Intaglio Imps manual control

If IntaglioImpsControl% = 1 Then

If IntaglioImpsManual% = 0 Then

IntaglioImpsManual% = 1

frmP00Intaglio!Label19.Visible = True

Else

IntaglioImpsManual% = 0

frmP00Intaglio!Label19.Visible = False

End If

End If

End Sub

Public Sub Numbers1\_Man\_Toggle()

'Numbering shaft 1 manual control

If Numbering\_1\_on% = 0 Then

Numbering\_1\_on% = 1

frmPressrun!Label31.Visible = True

Else

Numbering\_1\_on% = 0

frmPressrun!Label31.Visible = False

End If

End Sub

Public Sub Numbers2\_Man\_Toggle()

If Numbering\_2\_on% = 0 Then

Numbering\_2\_on% = 1

frmPressrun!Label33.Visible = True

Else

Numbering\_2\_on% = 0

frmPressrun!Label33.Visible = False

End If

End Sub

Public Sub IntWipe\_Man\_Toggle()

'Intaglio Wiping manual control

If IntaglioWipeControl% = 1 Then

If IntaglioWipeManual% = 0 Then

IntaglioWipeManual% = 1

frmP00Intaglio!Label16.Visible = True

Else

IntaglioWipeManual% = 0

Module1 - 19

```

        frmP00Intaglio!Label16.Visible = False
    End If
End If
VarChangeFlag%(40) = 1

```

End Sub

Public Sub IntPrewipe\_Man\_Toggle()

```

'Intaglio Prewipe manual control
If IntaglioPrewipeControl% = 1 Then
    If IntaglioPrewipeManual% = 0 Then
        IntaglioPrewipeManual% = 1
        frmP00Intaglio!Label22.Visible = True
    Else
        IntaglioPrewipeManual% = 0
        frmP00Intaglio!Label22.Visible = False
    End If
End If

```

End Sub

Public Sub Inker1nip\_Man\_Toggle()

```

'Intaglio Inker #1 nip control
If IntaglioInker1Control% = 1 Then
    If IntaglioInker1nip% = 0 Then
        IntaglioInker1nip% = 1
        frmP00Intaglio!Label27.Visible = True
    Else
        IntaglioInker1nip% = 0
        frmP00Intaglio!Label27.Visible = False
    End If
End If

```

End Sub

Public Sub Inker2nip\_Man\_Toggle()

```

'Intaglio Inker #2 nip control
If IntaglioInker2Control% = 1 Then
    If IntaglioInker2nip% = 0 Then
        IntaglioInker2nip% = 1
        frmP00Intaglio!Label45.Visible = True
    Else
        IntaglioInker2nip% = 0
        frmP00Intaglio!Label45.Visible = False
    End If
End If

```

End Sub

Public Sub Inker1run\_Man\_Toggle()

```

'Intaglio Inker #1 fountain control
If IntaglioInker1Control% = 1 Then
    If IntaglioInker1Manrun% = 0 Then
        IntaglioInker1Manrun% = 1
        frmP00Intaglio!Label26.Visible = True
    Else
        IntaglioInker1Manrun% = 0
        frmP00Intaglio!Label26.Visible = False
    End If
End If

```

End Sub

Public Sub Inker2run\_Man\_Toggle()

```

'Intaglio Inker #2 fountain control
If IntaglioInker2Control% = 1 Then
    If IntaglioInker2Manrun% = 0 Then

```

Module1 - 20

```

        IntaglioInker2Manrun% = 1
        frmP00Intaglio!Label41.Visible = True
    Else
        IntaglioInker2Manrun% = 0
        frmP00Intaglio!Label41.Visible = False
    End If
End If

End Sub
Public Sub IntInker1_Init_Off()

    'Intaglio Inker #1 mode: Off
    If MachineRunning% = 0 Then
        IntaglioInker1Control% = 0
        frmP00Intaglio!Label42.Visible = True
        frmP00Intaglio!Label43.Visible = False
        frmP00Intaglio!Label44.Visible = False
        VarChangeFlag%(34) = 1
        IntaglioInker1nip% = 0
        IntaglioInker1Manrun% = 0
        frmP00Intaglio!Label26.Visible = False
        frmP00Intaglio!Label27.Visible = False
        frmP00Intaglio!Command12.Enabled = False
        frmP00Intaglio!Command32.Enabled = False
    End If

End Sub
Public Sub IntInker2_Init_Off()

    'Intaglio Inker #2 mode: Off
    If MachineRunning% = 0 Then
        IntaglioInker2Control% = 0
        frmP00Intaglio!Label34.Visible = True
        frmP00Intaglio!Label33.Visible = False
        frmP00Intaglio!Label32.Visible = False
        VarChangeFlag%(35) = 1
        IntaglioInker2nip% = 0
        IntaglioInker2Manrun% = 0
        frmP00Intaglio!Label41.Visible = False
        frmP00Intaglio!Label45.Visible = False
        frmP00Intaglio!Command36.Enabled = False
        frmP00Intaglio!Command37.Enabled = False
    End If

End Sub

Public Sub FlexoImps_Man_Toggle()

    'Flexo Imps manual control
    If FlexoImpsControl% = 1 Then
        If FlexoImpsManual% = 0 Then
            FlexoImpsManual% = 1
            frmP00Flexo!Label19.Visible = True
        Else
            FlexoImpsManual% = 0
            frmP00Flexo!Label19.Visible = False
        End If
    End If

End Sub

```

```

NO'Program: Proofing press-processor 1
NO'Module : P00_1.dmc
NO'Version: V1.0
NO'Edit : -01
NO'Creation: 22-Sep-00
NO'Edit date: 01-Nov-01
NO'Author: Joseph B. Schutte III
NO'      Industrial Light&Motion, Inc.
NO'      2170 Van Blaricum Rd.
NO'      Cincinnati OH 45233 USA
NO'
NO'Copyright (C) 2000-2002
NO'Industrial Light & Motion, Inc.
NO'
NO'Revision History
NO'
NO'Ver/Ed   Date       Who   Reason
NO'-01 16-Oct-01 JBS Remap reg ctrls
NO'
NO'*****
NO'* Powerup / Initialization here *
NO'*****
#AUTO
ST XYZW;NO 'Stop all Axis'
AM XYZW
MO XYZW
NO [ * Define/Init variables * ]
DM AxisX[10];DM AxisY[10];DM AxisZ[10]
DM AxisW[10];DM AxisDXa[10]
DM AxisDXb[10]
BiasA=1.000; BiasB=1.000
BiasC=1.000; BiasD=1.000
CtWghtI=1000; CtWghtJ=1000
CtWghtK=1000; CtWghtL=1000
CutPerr=0; NumPerr=0; FlexPerr=0
FlexCreq=0; DieCreq=0; NO [Init flags]
Flexoffs=0; Dieoffs=0; NO [Init offsets]
NumRoffs=0; CutRoffs=0
FlexoffA=0; DieoffsA=0; NO [Init Acums]
NumoffsA=0; CutoffsA=0; NumSH=0; CutSH=0
FlexSH=0; Mictwght=1; DieEna=0
NumEna=0; CutEna=0; JogSpeed=20
LinSpeed=20; NO [set line speed]
Lsysmode=0; NO [Last scan SysMode]
Lsysrun=0; NO [Last scan SysRun]
ParamOK=0; SysCntr=0; SysMode=0; SysRun=0
SaveI=0; NO [Powerdown save reg]
SysState=0

#PARCLR
JP #PARCLR,ParamOK=0
NO 'Wait here till Host sets ParamOK
XQ #CREG,1; NO [Launch cutoff ctrl]
XQ #ABSREG,2; NO [ABS update routine]
XQ #SUPPORT,3; NO [Launch aux task]
#MAIN

NO [ * Beginning of Executive Loop * ]
#MAIN00
JP #MAIN50, SysRun=4; NO [Reset faults]

```



```

JP #MAIN80, SysRun=6; NO [Drives Reset]
JP #MAIN10, SysRun=0; NO [Stop seq]
JP #MAIN60, SysRun=1; NO [Cal sequence]
JP #MAIN30, SysRun=2; NO [Run request]
JP #MAIN40, SysRun=3; NO [Jamup Stop]
JP #MAIN70, SysRun=5; NO [Jog seq]
JP #MAIN90, SysRun=9; NO [Powerdown]
JP #MN100, SysRun=10; NO [Set Sysmode]
JP #MAIN00; NO [Unrecognized command]

```

```

#MAIN10
NO '*****
NO '* System Normal Stop SysRun=0 *
NO '*****
NO
JP #MAIN00, SysState=0; NO [stopped]
JP #MAIN19, (SysState=1)|(SysState=3)
JP #MAIN19, (SysState=4)|(SysState=6)
JP #MAIN19, SysMode<>0; NO [Not Flexo]
NO [No-op states 1,3,4,6]
ST X;AMX;NO [Stop Master]
#MAIN19
SysState=0
JP #MAIN00

```

```

#MAIN30
NO '*****
NO '* RUN Mode SysRun=2 *
NO '*****
NO

```

```

#MAIN40
NO '*****
NO '* System Jamup Stop Mod SysRun=3 *
NO '*****
NO
JP #MAIN00, SysState=3; NO [stopped?]
JP #MAIN49, (SysState=1)|(SysState=0)
JP #MAIN49, (SysState=4)|(SysState=6)

```

```

JP #MAIN49, SysMode<>0
STX; AMX; MO XYZW; NO [Stop Master]
JP #MAIN49
#MAIN49
SysState=3
JP #MAIN00

#MAIN50
NO '*****
NO '* Fault Reset Mode      SysRun=4  *
NO '*****

JP #AUTO

#MAIN60
NO '*****
NO '* System Cal Mode      SysRun=1    *
NO '*****

SysState=1; NO [Calibrate mode]
Flexoffs=0; Dieoffs=0; NO[Init offsets]
NumRoffs=0; CutRoffs=0
FlexoffA=0; DieoffsA=0; NO[ Init Acums]
NumoffsA=0; CutoffsA=0
NO [Select entry point]
#MAIN61
JP #MAIN68, SysRun<>1
JP #MAIN62, FlexCreq<>0
STX; AMX; NO [Stop in case running]
JP #MAIN65
#MAIN62; NO [Jog Flexo axis]
JGX=ProdHos*12*CtWghtI/60; BGX
#MAIN65
JP #MAIN66, DieCreq<>0
ST Y; AM YZW; NO [Stop axis']
JP #MAIN61

```

```

#MAIN70
NO '*****
NO '* JOG Mode           SysRun=5      *
NO '*****

SysState=5;NO [Set Jog mode status]
JP #MAIN75, SysMode<>0
JGX=JogSpeed*12*CtWghtI/60
BG X; NO [Start line]

#MAIN73
GR ,(CtWghtJ/CtWghtI)*BiasB
GR ,(CtWghtK/CtWghtI)*BiasC
GR ,(CtWghtL/CtWghtI)*BiasD
JGX=JogSpeed*12*CtWghtI/60
JP #MAIN00, SysRun<>5
JP #MAIN73

#MAIN75
GR (CtWghtI/Mictwght)*BiasA
GR ,(CtWghtJ/Mictwght)*BiasB
GR ,(CtWghtK/Mictwght)*BiasC
GR ,(CtWghtL/Mictwght)*BiasD
JP #MAIN00, SysRun<>5
JP #MAIN75

#MAIN80
NO '*****
NO '* Servo Loop Reset Mode SysRun=6 *
NO '*****

SysState=0
ST*;NO [Stop all drives]
MO XY; NO [Inhibit drives]
JP #MAIN00

#MAIN90
NO '*** Powerdown mode - SysRun=9 ***
SaveI= TPX; SysState=9
#MAIN91
JP #MAIN91, SysRun=9
DEX=SaveI
JP #MAIN00

#MN100
NO '*****
NO '* Set System Mode     SysRun=10    *
NO '*****

```



EN

```

NO '*****
NO '* Logic for *
NO '* Diecut Auto Register *
NO '*****
#CREG
Cregrefm=0; cmks=0; Lcreg=0
#CREG00A; AxisY[8]=0
#CREG00
AxisDXb[2]= TDX; JS #TAXDX1
posH=AxisDXb[4]
JP #CREG002, _ALY=0; NO [Mark?]
JP #CREG001, posH<CutLwin
JP #CREG001, posH>CutUwin
JP #CREG00
#CREG001
cmks=0; NO [Out of window]
JP #CREG00
#CREG002; NO [Have mark]
AxisY[2]=_RLY; JS #TAXY
Creg0=AxisY[4]
ALY
JP #CREG00, cmks>0
JP #CREG00, posH<CutLwin
JP #CREG00, posH>CutUwin
JP #CREG00A, CutRauto=0

```

```

#CREG100
Cregrefm=posH
Creg0=(CutRoffs*CtWghtJ)-Creg0
JP #CREG102, (Creg0*-1)>(CPRJ/2)
JP #CREG103, Creg0>(CPRJ/2)
JP #CREG104
#CREG102; Creg0=Creg0+CPRJ
JP #CREG104
#CREG103; Creg0=Creg0-CPRJ
#CREG104
CutPerr=Creg0/CtWghtJ
JP #CREG105, Creg0>2500; NO Limiter
JP #CREG105, (Creg0*-1) <-2500
JP #CREG109
#CREG105
JP #CREG109, Lcreg>2500
JP #CREG109, (Lcreg*-1) <-2500
Lcreg=Creg0; Creg0=0
JP #CREG110
#CREG109; Lcreg=Creg0
#CREG110
JP #CREG115, (Creg0*-1)>CutMEmax
JP #CREG116, Creg0>CutMEmax
JP #CREG117
#CREG115; Creg0=CutMEmax*-1
JP #CREG117
#CREG116; Creg0=CutMEmax
#CREG117
Creg0=Creg0*CutMEgn*-1
AMY; IPY=Creg0; cmkslast=Creg0
AxisY[8]=AxisY[8]+Creg0
cmks=1
JP #CREG00
EN

```

```

NO '*****
NO '* Logic for      *
NO '* Support variables *
NO '*****
#SUPPORT
NO [Calculate the actual linespeed]
ALspeed=_TVX/CtWghtI*5;
WT 10
JP #SUPPORT

```

```

NO '*****
NO '* Logic for      *
NO '* ABS register routine *
NO '*****
#ABSREG
WT 1000
JP #ABSREG, (SysRun<2)
NO [Update Flexo position]
JP #REGUP01, FlexSH=0
AxisDXa[2]=_TDX; AxisX[2]=_TPX
JS #TAXDX0; JS #TAXX
refpos=AxisDXa[4]; slpos=AxisX[4]
refpos=refpos/Mictwght
sloff=Flexoffs; ctwght=CtWghtI
cprs=CPRI

```

```

JS #ECALC
AMX; IP poserr02*AxisX[7]
FlexPerr=poserr03
#REGUP01
NO [Update Diecutter pos]
#REGUP02
NO [Update Diecut pull pos]
#REGUP03
NumPerr=0
NO [Update Sheeter position]
JP #ABSREG

#ECALC; NO [Pos error calculator]
slpos=slpos/ctwght
poserr02=slpos+sloff
poserr03=refpos-poserr02; NO [Error]
poserr04=poserr03*ctwght
JP #ECALC02, (poserr04*-1)>(cprs/2)
JP #ECALC03, poserr04>(cprs/2)
JP #ECALC04
#ECALC02; poserr04=poserr04+cprs
JP #ECALC04
#ECALC03; poserr04=poserr04-cprs
#ECALC04
poserr03=poserr04/ctwght
JP #ECALC05, poserr04<(Maxerr*-1)
JP #ECALC06, poserr04>Maxerr
JP #ECALC07
#ECALC05; poserr04=Maxerr*-1
JP #ECALC07
#ECALC06; poserr04=Maxerr
#ECALC07
poserr02=poserr04
EN

NO [***Counter mgmt Code***]

NO [ Array definitions ]
NO [ 0 - Active mod counter value ]
NO [ 2 - Hardware counter value ]
NO [ 3 - Last scan counter value ]
NO [ 4 - Calc'ed actual position ]
NO [ 6 - Masked counter value ]
NO [ 7 - Correction gain ]
NO [ 8 - Correction accumulator ]
NO [ 9 - Temp scratch register ]

#TAXDX0
AxisDXa[9]=AxisDXa[2]&$0FFFFFFF
AxisDXa[6]=AxisDXa[9]
AxisDXa[9]=AxisDXa[9]-AxisDXa[3]
JP #TAXDX02, AxisDXa[9]>8388608
JP #TAXDX03, AxisDXa[9]<-8388608
JP #TAXDX04
#TAXDX02
AxisDXa[9]=AxisDXa[9]-16777216
JP #TAXDX04
#TAXDX03
AxisDXa[9]=AxisDXa[9]+16777216
#TAXDX04; AxisDXa[3]=AxisDXa[6]

```

```

AxisDXa[0]=AxisDXa[0]+AxisDXa[9]
#TAXDX05
JP #TAXDX06, AxisDXa[0]>(CPRDX-1)
JP #TAXDX07, AxisDXa[0]<0
JP #TAXDX08
#TAXDX06
AxisDXa[0]=AxisDXa[0]-CPRDX
JP #TAXDX05
#TAXDX07
AxisDXa[0]=AxisDXa[0]+CPRDX
JP #TAXDX05
#TAXDX08; AxisDXa[4]=AxisDXa[0]
EN

```

```

#TAXDX1
AxisDXb[9]=AxisDXb[2]&$0FFFFFFF
AxisDXb[6]=AxisDXb[9]
AxisDXb[9]=AxisDXb[9]-AxisDXb[3]
JP #TAXDX12, AxisDXb[9]>8388608
JP #TAXDX13, AxisDXb[9]<-8388608
JP #TAXDX14
#TAXDX12
AxisDXb[9]=AxisDXb[9]-16777216
JP #TAXDX14
#TAXDX13
AxisDXb[9]=AxisDXb[9]+16777216
#TAXDX14; AxisDXb[3]=AxisDXb[6]
AxisDXb[0]=AxisDXb[0]+AxisDXb[9]
#TAXDX15
JP #TAXDX16, AxisDXb[0]>139999
JP #TAXDX17, AxisDXb[0]<0
JP #TAXDX18
#TAXDX16
AxisDXb[0]=AxisDXb[0]-140000
JP #TAXDX15
#TAXDX17
AxisDXb[0]=AxisDXb[0]+140000
JP #TAXDX15
#TAXDX18; AxisDXb[4]=AxisDXb[0]
EN

```

```

#TAXX
AxisX[9]=AxisX[2]&$0FFFFFFF
AxisX[6]=AxisX[9]
AxisX[9]=AxisX[9]-AxisX[3]
JP #TAXX02, AxisX[9]>8388608
JP #TAXX03, AxisX[9]<-8388608
JP #TAXX04
#TAXX02
AxisX[9]=AxisX[9]-16777216
JP #TAXX04
#TAXX03
AxisX[9]=AxisX[9]+16777216
#TAXX04; AxisX[3]=AxisX[6]
AxisX[0]=AxisX[0]+AxisX[9]
#TAXX05
JP #TAXX06, AxisX[0]>(CPRI-1)
JP #TAXX07, AxisX[0]<0
JP #TAXX08
#TAXX06

```

```

AxisX[0]=AxisX[0]-CPRI;JP #TAXX05
#TAXX07
AxisX[0]=AxisX[0]+CPRI;JP #TAXX05
#TAXX08; AxisX[4]=AxisX[0]
EN

```

```

#TAXY
AxisY[9]=AxisY[2]&$0FFFFFFF
AxisY[6]=AxisY[9]
AxisY[9]=AxisY[9]-AxisY[3]
JP #TAXY02, AxisY[9]>8388608
JP #TAXY03, AxisY[9]<-8388608
JP #TAXY04
#TAXY02
AxisY[9]=AxisY[9]-16777216
JP #TAXY04
#TAXY03
AxisY[9]=AxisY[9]+16777216
#TAXY04; AxisY[3]=AxisY[6]
AxisY[0]=AxisY[0]+AxisY[9]
#TAXY05
JP #TAXY06, AxisY[0]>(CPRJ-1)
JP #TAXY07, AxisY[0]<0
JP #TAXY08
#TAXY06
AxisY[0]=AxisY[0]-CPRJ;JP #TAXY05
#TAXY07
AxisY[0]=AxisY[0]+CPRJ;JP #TAXY05
#TAXY08; AxisY[4]=AxisY[0]
EN

```

```

#TAXW
AxisW[9]=AxisW[2]&$0FFFFFFF
AxisW[6]=AxisW[9]
AxisW[9]=AxisW[9]-AxisW[3]
JP #TAXW02, AxisW[9]>8388608
JP #TAXW03, AxisW[9]<-8388608
JP #TAXW04
#TAXW02
AxisW[9]=AxisW[9]-16777216
JP #TAXW04
#TAXW03
AxisW[9]=AxisW[9]+16777216
#TAXW04; AxisW[3]=AxisW[6]
AxisW[0]=AxisW[0]+AxisW[9]
#TAXW05
JP #TAXW06, AxisW[0]>65535
JP #TAXW07, AxisW[0]<0
JP #TAXW08
#TAXW06
AxisW[0]=AxisW[0]-65536;JP #TAXW05
#TAXW07
AxisW[0]=AxisW[0]+65536;JP #TAXW05
#TAXW08; AxisW[4]=AxisW[0]
EN

```

```

NO      'Program      : Press 00 Processor 0
NO      'Module       : P00_0.dmc
NO      'Version      : V1.0
NO      'Edit         : -01
NO      'Creation     : 13-Sep-00
NO      'Edit date    : 05-Nov-00
NO      'Author       : Joseph B. Schutte III
NO      '              : Industrial Light & Motion, Inc.
NO      '              : 2170 Van Blaricum Rd.
NO      '              : Cincinnati OH USA
NO      '
NO      'Copyright (C) 1998-2002 Industrial Light & Motion, Inc.
NO      '
NO      'Revision History
NO      '
NO      'Ver/Ed      Date      Who      Reason
NO      '
NO      'Sensor assignments:
NO      'VB1:IN2(LY) VB2:IN3(LZ) VB3:IN5(LE) VB4:IN6(LF)
NO      'Intaglio marks:IN8(LH) Cutoff marks:IN1(LX)

NO      '*****
NO      '*          Powerup / Initialization here          *
NO      '*          *                                       *
NO      '*****
#AUTO
ST XYZWEFGH;NO 'Stop all Axis'
AM XYZWEFGH
MO XYZWEFGH

NO [ * Define/Init variables * ]
NO
NO [Axis Acc/Dec parameters from host]
NO AxisAAC, AxisBAC, AxisCAC, AxisDAC
NO AxisEAC, AxisFAC, AxisGAC, AxisHAC
ALspeed=0; DieRauto=0; arFlexoB=1.000
CalState=0;NO [Calibrate mode]
DieMKpos=0
MSIJreq=0; MSWIreq=0; MSWJreq=0; NO [Requests for moves]
ChiEna=0; PIEna=0; POEna=0; SHEna=0; WIEna=0
EstopAC=200000
FlexCreq=0; IntCreq=0; DieCreq=0; NO [Init req flags]
Iposerr=0; Dposerr=0
JogSpeed=20
LastABSa=0; LastABSg=0
LinSpeed=20; NO [set line speed]
LockFlt=0; NO [Inhibit sys on fault]
Lsysmode=0; NO [Last scan SysMode]
Lsysrun=0; NO [Last scan SysRun]
Mlctwght=1000
NormAC=7500; NO [Acc/Dec ramp norm run]
ParamOK=0
RevCTR=0; lcntr=TPH; NO [Init the system counter]
SysCntr=0; NO [Num of imp since pwrap]
SysMode=0
SysRun=0;
SysStat=0; NO [System status ]
SysState=0; NO [Current system state]

```

```

Testvar=0; NO [Debug variable]
VBregCTR=0; VBdieCTR=0; RevCTR=0; IntMarkC=0; DieMarkC=0; NO [Reg vars]
Wigain=1; Wogain=1
WipeMode=0; NO [Wiping transport mode]
ProdCal=0; WipeCal=0

```

```

NO '* Insystem variables *
NO AxisAAC=2000
NO AxisBAC=2000000
NO AxisCAC=2000000
NO AxisDAC=2000000
NO AxisEAC=2000000
NO AxisFAC=2000000
NO AxisGAC=2000000
NO AxisHAC=2000000

```

```

#PARCLR
JP #PARCLR, ParamOK=0
NO 'Wait here till Host sets ParamOK

```

```

#MAIN
XQ #SHUTM, 1; NO [Start Shuttle transport]
XQ #WIPEM, 2; NO [Start wiping task]
XQ #DIEM, 3; NO [Start diecut register task]

```

```

NO [ * Beginning of Executive Loop * ]
#MAIN00

```

```

JP #MAIN50, SysRun=4; NO [Reset faults]
JP #MAIN80, SysRun=6; NO [Drives Reset]
JP #MAIN00, LockFlt>0; NO [Locked out?]
JP #MAIN10, SysRun=0; NO [Stop seq]
JP #MAIN30, SysRun=2; NO [Run request]
JP #MAIN40, SysRun=3; NO [Jamup Stop]
JP #MAIN70, SysRun=5; NO [Jog seq]
JP #MN100, SysRun=10; NO [Set Sysmode]
JP #MAIN00; NO [Unrecognized command]

```

```

#MAIN10
NO '*****
NO '* System Normal Stop SysRun=0 *
NO '* *
NO '*****
NO
SysState=SysRun; NO [Update state]
JS #COUNTER; NO [Update system counter]
JP #MAIN15, SysMode<>0; NO [Combi?]
JP #MAIN00

```

```

#MAIN15
GR (CtWghtA/MIctwght)*BiasA*-1
GR , , , (CtWghtD/MIctwght)*BiasD*-1
GR , , , , (CtWghtG/MIctwght)*BiasG*-1
JP #MAIN00

```

```

#MAIN30
NO '*****

```

```

NO '* RUN Mode                      SysRun=2 *
NO '*
NO '*****
NO
SysState=SysRun;NO [Update state]
JP #MAIN35, SysMode<>0; NO [Combi?]
#MAIN31
JS #COUNTER; NO [Update system counter]
JP #MAIN00, SysRun<>2
JP #MAIN31

#MAIN35
GR (CtWghtA/Mictwght)*BiasA*-1
JS #COUNTER; NO [Update system counter]
JP #MAIN00, SysRun<>2
JP #MAIN35

#MAIN40
NO '*****
NO '* System Jamup Stop Mod SysRun=3 *
NO '*
NO '* EstopAC controls decel ramp. *
NO '*****
NO
SysState=SysRun;NO [Update state]
JP #MAIN45, SysMode<>0; NO [Combi?]
#MAIN42
JP #MAIN00, SysRun<>3
JP #MAIN42

#MAIN45
GR (CtWghtA/Mictwght)*BiasA*-1
GR ,,, (CtWghtD/Mictwght)*BiasD*-1
GR ,,,,, (CtWghtG/Mictwght)*BiasG*-1
JP #MAIN00, SysRun<>3
JP #MAIN45

#MAIN50
NO '*****
NO '* Fault Reset Mode          SysRun=4 *
NO '*
NO '* Called on reset. *
NO '*
NO '*****

JP #AUTO

#MAIN70
NO '*****
NO '* JOG Mode                      SysRun=5 *
NO '*
NO '*****

SysState=5;NO [Set Jog mode status]
JP #MAIN75, SysMode<>0
#MAIN71
JS #COUNTER; NO [Update system counter]
JP #MAIN00, SysRun<>5
JP #MAIN71

```



```

#MAIN75
GR (CtWghtA/Mictwght)*BiasA*-1
GR ,,, (CtWghtD/Mictwght)*BiasD*-1
GR ,,,,, (CtWghtG/Mictwght)*BiasG*-1
JS #COUNTER; NO [Update system counter]
JP #MAIN00, SysRun<>5
JP #MAIN75

#MAIN80
NO *****
NO '* Servo Loop Reset Mode   SysRun=6 *
NO '*
NO *****

SysState=0
CalState=0; NO [Cancel calibrate status]
SysStat=0
ST*; NO [Stop all drives]
MO XY; NO [Inhibit drives]
JP #MAIN00

#MN100
NO *****
NO '* Set System Mode       SysRun=10 *
NO '*
NO *****

SysState=100; NO [Set Combo mode]
ST XYZWEFGH; NO [Stop everything]
MO XYZWEFGH
JP #MN119, SysMode=0; NO [ Flexo only? ]
#MN110
GA DX,,,DX,,,DX; NO [Set reference]
GR (CtWghtA/Mictwght)*BiasA*-1
GR ,,, (CtWghtD/Mictwght)*BiasD*-1
GR ,,,,, (CtWghtG/Mictwght)*BiasG*-1
JP #MN111, PIEna=0; NO [Product Infeed]
SH X
#MN111
JP #MN112, SHEna=0; NO [Shuttle rolls]
SH YZ
#MN112
JP #MN113, POEna=0; NO [[Product outfeed]
SH W
#MN113
JP #MN119, ChiEna=0; NO [Chill rolls]
SH G
#MN119
SysRun=100
JP #MAIN00

EN

#COUNTER
NO [ *** System Counter support *** ]
cntr=TPH; NO [Get current master position]
JP #CNTR001, cntr>10000; NO [Are we in the window?]
JP #CNTR001, lcntr<130000
RevCTR=RevCTR+1

```

```

#CNTR001
lcntr=cntr
EN

NO '*****
NO '*          Logic for Product Shuttle System          *
NO '*          *
NO '*****
#SHUTM
TrigCam=0; PMode=0; vb1P=0

#SHUT00
JS #INITCAM, TrigCam<>0; NO [Cam init request]
JP #SHUT000, SysRun=0; NO [Stop request]
JP #SHUT100, SysRun=1; NO [Product calibration req]
JP #SHUT200, SysRun=2; NO [System RUN]
JP #SHUT500, SysRun=5; NO [System JOG]
JP #SHUT1000, SysRun=10; NO [Initialization seq]
JP #SHUT00; NO [Unsupported command]

#SHUT000; NO [ Stop Sequence ]
PMode=0
JP #SHUT00

#SHUT100; NO [...Calibrate Sequence - Product...]

PMode=1; ProdCal=0
ST ADG; AM ADG; NO [Disengage PI servo]
GAG=W; GRG=(CtWghtG/CtWghtD)*BiasG

JP #SHUT105, @IN[2]<>0; NO [Jump if VB1 full]
NO [...PI (VB1) low on paper...]
JP #SHUT101, SysMode<>2; NO [Flexo only combi mode]
FlexCreq=1; WT 50; NO [Request Flexo move]
#SHUT101
JGA=ProdHos*CtWghtA/5; BGA
#SHUT102
JP #SHUT180, SysRun<>PMode; NO [Abort?]
JP #SHUT102, @IN[2]=0
JP #SHUT103, SysMode<>2; NO [Only need Flexo in combi]
FlexCreq=0; WT50; NO [Cancel request]
#SHUT103
JGA=1; NO [Stop drive]
JP #SHUT110; NO [PI box calibrated]

#SHUT105; NO [...PI (VB1) too much paper...]
ST ADG; AM ADG

IntCreq=1; NO [Start Main drive]
#SHUT106
JP #SHUT180, SysRun<>PMode
JP #SHUT106, @IN[2]<>0
IntCreq=0; NO [Stop Intaglio drive]
JP #SHUT110; NO [PI box calibrated]

#SHUT110; NO [VB2 / PO box calibration]

```

```

JP #SHUT115, @IN[3]<>0; NO [Jump if VB2/WO box full]
GRD=0; GRG=0; STA; AMA; NO [Inhibit Chill and PO drives]
IntCreq=1; WT70; NO [Starting Flexo and Int main drives]
FlexCreq=1; WT50; NO [...VB2 low on paper...]
JGA=ProdHos*CtWghtA/5; BGA
#SHUT111
JP #SHUT180, SysRun<>PMode
JP #SHUT111, @IN[3]=0
IntCreq=0; WT50; FlexCreq=0; WT50; NO [Stop drives]
STA; AMA; NO [Stop PI servo]
JP #SHUT120; NO [...VB2 calibrated..]

#SHUT115; NO [VB2 has too much paper]
GAG=W; GRG=(CtWghtG/CtWghtD)*BiasG
JGD=ProdHos*CtWghtD/5
DieCreq=1; WT50; BGD; NO [Request Diecutting units to run]
#SHUT116
JP #SHUT180, SysRun<>PMode
JP #SHUT116, @IN[3]<>0
DieCreq=0; WT50; STD; AMD; NO [We're there - stop]

#SHUT120; NO [Calibrated and waiting]
ProdCal=255
#SHUT121
JP #SHUT121, SysRun=PMode
JP #SHUT190

#SHUT180; NO [Bailout of calibration]
IntCreq=0; FlexCreq=0; DieCreq=0; NO [Stop all Req's]
ST ADG; AM ADG
ProdCal=0
JP #SHUT190

#SHUT200; NO [...Run Mode - Product paper...]
PMode=2
#SHUT201
JP #SHUT00, PMode<>SysRun
JP #SHUT201, SysMode=0; NO [Flexo only mode?]
JP #SHUT250, IntrAuto<>0; NO [Auto (mark) register mode?]

NO [VB regulator mode - Manual]
VBregCTR=RevCTR; NO [First cycle sync counter]
arFlexoB=1.000; NO [Reset auto reg Flexo bias to null]
#SHUT210
JP #SHUT00, PMode<>SysRun
JP #SHUT201, IntrAuto<>0
vb1Plast=vb1P; NO [Achieve the previous VB sensor read]
vb1P=@IN[2]; NO [Get ther current state]
JP #SHUT211, (vb1P=0)&(vb1Plast<>0); NO [Falling edge?]
JP #SHUT212, ((VBregCTR+2)<RevCTR)&(vb1P=0)
JP #SHUT213, ((VBregCTR+2)<RevCTR)&(vb1P<>0)
JP #SHUT210; NO [Loop...]

```

```

#SHUT211
vbpos1=TPH; NO [Read the reference encoder]
VB1sampc=vbpos1
Iposerr=VB1nullP-vbpos1; NO [This is the pos error]
JP #SHUT213, Iposerr<(IntVBmax*-1); NO [Error out of - range]
JP #SHUT212, Iposerr>IntVBmax; NO [Error out of + range]
ipos001=Iposerr; JP #SHUT214; NO [Error in range]
#SHUT212
ipos001=IntVBmax; JP #SHUT214
#SHUT213
ipos001=IntVBmax*-1; JP #SHUT214
#SHUT214
ipos001=ipos001*IntVBgn; NO [Adjust for error gain]
SH1corr=ipos001
#SHUT215; NO [Wait for end of print cycle]
JP #SHUT00, PMode<>SysRun

```

```

JP #SHUT210

```

```

NO [Intaglio Auto register mode]
#SHUT250
ALH; NO [Intaglio mark latch]
#SHUT251
NO **** JS #SHUT270; NO [VB1 auto mode support]
JP #SHUT00, PMode<>SysRun
JP #SHUT201, IntRauto=0
JP #SHUT251, ALH<>0
JP #SHUT250, (_RLH<IntLwin)|(_RLH>IntUwin)
VB1sampc=RLH

```

```

ipos001=Iposerr; JP #SHUT254; NO [Error in range]
#SHUT252
ipos001=IntMEmax; JP #SHUT254
#SHUT253
ipos001=IntMEmax*-1; JP #SHUT254
#SHUT254
ipos001=ipos001*IntMEgn; NO [Adjust for error gain]
SH1corr=ipos001
#SHUT255; NO [Wait till end of print cycle]
JS #SHUT270; NO [VB1 auto mode support]
JP #SHUT00, PMode<>SysRun

```

```

JP #SHUT250

```

```

NO [Intaglio Auto Reg VB1 support]
#SHUT270
vb1Plast=vb1P; vb1P=@IN[2]
JP #SHUT271, (vb1P<>0)&(vb1Plast=0); NO [Rising edge]
JP #SHUT275, ((VBregCTR+2)<RevCTR)&(vb1P<>0)
JP #SHUT276, ((VBregCTR+2)<RevCTR)&(vb1P=0)
JP #SHUT279; NO [No activity]

```

```
#SHUT271; vbpos1=TPH; VBlautoE=VBlnullA-vbpos1
JP #SHUT272, (VBlautoE*-1)>(CPRH/2); NO [Neg rollover?]
JP #SHUT273, VBlautoE>(CPRH/2); NO [Pos rollover?]
JP #SHUT274
#SHUT272; VBlautoE=VBlautoE+CPRH; JP #SHUT274
#SHUT273; VBlautoE=VBlautoE-CPRH
#SHUT274
JP #SHUT276, VBlautoE<(arVBmax*-1)
JP #SHUT275, VBlautoE>arVBmax
JP #SHUT277
#SHUT275; VBlautoE=arVBmax; JP #SHUT277
#SHUT276; VBlautoE=arVBmax*-1
, #SHUT277

#SHUT279
EN

#SHUT500; NO [...Jog Mode - Product paper...]
PMode=5
#SHUT501
JP #SHUT00, PMode<>SysRun
JP #SHUT501

#SHU1000; NO [...Initialization sequence - Product paper...]
PMode=10
#SHU1001
JP #SHUT00, PMode<>SysRun
JP #SHU1001

.#INITCAM; NO [Initialize the SHUTTLE cam]
```

EN

```

NO *****
NO '*          Logic for Wiping System          *
NO '*          *
NO *****
#WIPEM
WMode=0; NO [WipeMode follows SysRun ex. offline]
WipeCal=0; vb3P=0; vb4P=0; vb3rcctr=0; vb4rcctr=0

NO [ * Beginning of Wiping Loop * ]
#WIPE00
JP #WIPE990, WipeMode=0; NO [Are we offline?]
JP #WIPE01, WMode<>99; NO [Do we need to powerup drives?]
SH EF
WMode=0

#WIPE01
JP #WIPE000, SysRun=0; NO [Stop seq]
JP #WIPE200, SysRun=2; NO [Run request]
JP #WIPE800, SysRun=8; NO [Jog seq]
JP #WIPE900, SysRun=9; NO [Calibrate mode]
JP #WIPE00; NO [Unrecognized command]

#WIPE000; NO [Stop Sequence]
WMode=0

#WIPE001
JP #WIPE00, SysRun<>WMode
JS #WIPE960; NO [Update the WI - WO speed loop]
JP #WIPE001, ALspeed>WautoonS; NO [Down to Wipe off speed?]
#WIPE006
w001= TPH
JP #WIPE007, ((w001+3000)>IPwipaut)&((w001-3000)<IPwipaut)
JS #WIPE960; NO [Update the WI - WO speed loop]
JP #WIPE006, ALspeed>5; NO [Wait if we haven't stopped yet]
#WIPE007
MSWiReq=0; NO [Wiping impressions off]
ST EF
#WIPE008
JP #WIPE008, (SysRun=WMode)&(WipeMode<>0) ; NO [Wait for mode to change]
JP #WIPE00

#WIPE200; NO [Run Sequence]
WMode=2
#WIPE201

```



```

JP #WIPE902, @IN[5]<>0; NO [VB3 full]
JGE=WipeHos*CtWghtE/5; NO [...Need paper...]
#WIPE901
JP #WIPE985, SysRun<>WMode; NO [Abort?]
JP #WIPE901, @IN[5]=0
JGE=1; NO [Stop drive]
JP #WIPE910; NO [WI box calibrated]

#WIPE902; NO [...WI (VB3) too much paper...]
MSWJreq=1; NO [Start wiping drive]
JGF=WipeHos*CtWghtF/5
#WIPE903
JP #WIPE985, SysRun<>WMode
JP #WIPE903, @IN[5]<>0
MSWJreq=0; JGF=1; NO [Stop wiping drive]
JP #WIPE910; NO [WI box calibrated]

#WIPE910; NO [VB4 / WO box calibration]
JP #WIPE912, @IN[6]<>0; NO [VB4 / WO box full]
MSWJreq=1; NO [Start Wiping and PI drives-we're low]
JGE=WipeHos*CtWghtE/5
#WIPE911
JP #WIPE985, SysRun<>WMode
JP #WIPE911, @IN[6]=0
MSWJreq=0; JGE=1; NO [Stop drives]
JP #WIPE920

#WIPE912; NO [VB4 has too much paper]
JGF=WipeHos*CtWghtF/5; NO [...Need paper...]
#WIPE913
JP #WIPE985, SysRun<>WMode
JP #WIPE913, @IN[6]<>0
JGF=1

#WIPE920
SPE=5000; SPF=5000
ST EF; AM EF
PRE=WVBcalof
PRF=WVBcalof
BG EF
AM EF; MSWJreq=0; NO [Wiping imp's off]
WipeCal=255
#WIPE921
JP #WIPE921, SysRun=WMode
JP #WIPE00

```

EN

#WIPE965; NO [...VB3...]



```

#WIPE966
Wposerr=Wposerr+CPRH; JP #WIPE968
#WIPE967
Wposerr=Wposerr-CPRH
#WIPE968
JP #WIPE969, Wposerr<(WipVBmax*-1)
JP #WIPE970, Wposerr>WipVBmax
Wpos001=Wposerr; JP #WIPE971
#WIPE969
Wpos001=WipVBmax*-1; JP #WIPE971
#WIPE970
Wpos001=WipVBmax
#WIPE971

```

EN

```

#WIPE975; NO [...VB4...]
JP #WIPE979, ((vb4rctr+2)<RevCTR)&(vb4P<0); NO [VB4 box FULL!]
JP #WIPE980, ((vb4rctr+2)<RevCTR)&(vb4P=0); NO [VB4 box EMPTY!]
WVBpos= TRH; Wposerr=VB4nullP-WVBpos; NO [This is the error]
JP #WIPE976, (Wposerr*-1)>(CPRH/2); NO [Negative rollover?]
JP #WIPE977, Wposerr>(CPRH/2); NO [Positive rollover?]
JP #WIPE978
#WIPE976
Wposerr=Wposerr+CPRH; JP #WIPE978
#WIPE977
Wposerr=Wposerr-CPRH
#WIPE978
JP #WIPE979, Wposerr<(WipVBmax*-1)
JP #WIPE980, Wposerr>WipVBmax
Wpos001=Wposerr; JP #WIPE981
#WIPE979
Wpos001=WipVBmax*-1; JP #WIPE981
#WIPE980
Wpos001=WipVBmax
#WIPE981

```

EN

```

#WIPE985; NO [Bailout of calibration]
ST EF; MSWJreq=0; MSWJreq=0
JP #WIPE00

```

```

#WIPE990; NO [Offline - disable drives]
JP #WIPE00, WMode=99
ST EF; MO EF
WMode=99
JP #WIPE00

```

EN

```

NO *****
NO '*      Logic for Diecutting Register control      *
NO '*      *
NO *****
#DIEM
vb2P=0
NO [...Run Mode - Product paper...]

#DIEM00

#DIEM201
JP #DIEM00, SysRun<>2
JP #DIEM201, SysMode=0; NO [Flexo only mode?]
JP #DIEM250, DieRauto<>0; NO [Auto (mark) register mode?].

NO [VB regulator mode - Manual]
VBdieCTR=RevCTR
#DIEM210
JP #DIEM00, SysRun<>2
JP #DIEM201, DieRauto<>0
vb2Plast=vb2P; NO [Archive the previous VB sensor read]
vb2P=@IN[3]; NO [Get ther current state]
JP #DIEM211, (vb2P<>0)&(vb2Plast=0); NO [Falling edge?]
JP #DIEM212, ((VBdieCTR+2)<RevCTR)&(vb2P<>0); NO [Box Outer limit?]
JP #DIEM213, ((VBdieCTR+2)<RevCTR)&(vb2P=0)
JP #DIEM210; NO [Loop...]
#DIEM211
vbpos2=_TPH; NO [Read the reference encoder]
VB2sampo=vbpos2
Dposerr=VB2nullP-vbpos2; NO [This is the pos error]
JP #DIEM213, Dposerr<(DieVBmax*-1); NO [Error out of - range]
JP #DIEM212, Dposerr>DieVBmax; NO [Error out of + range]
dpos001=Dposerr; JP #DIEM214; NO [Error in range]
#DIEM212
dpos001=DieVBmax; JP #DIEM214
#DIEM213
dpos001=DieVBmax*-1; JP #DIEM214
#DIEM214

JP #DIEM210

NO [Diecutter Auto register mode]
#DIEM250
dmks=0; DieMacum=0; DieMctr=0; DieWctr=0
#DIEM251
dmkslast=dmks; dmks=@IN[1]; NO [Get mark state]
JP #DIEM253, (dmks=0)&(dmkslast<>0); NO [Falling edge of mark?]
JP #DIEM00, SysRun<>2; NO [Line stop?]
JP #DIEM201, DieRauto=0; NO [Return to manual?]
JP #DIEM251

```

```
#DIEM253
posh= TPH; NO [Read position as quickly as possible]
JP #DIEM251, (posh<DieLwin)|(posh>DieUwin)
DieMKpos=posh; NO [Save for VB display]
Dposerr=(Dieoffs*Mictwght)-posh; NO [calculate the error]
DieMacum=DieMacum+Dposerr; DieMctr=DieMctr+1; NO [Get data]
JP #DIEM251, DieMctr<DieSave; NO [Enough data]
dpos001=DieMacum/DieSave; NO [Calc the average]
JP #DIEM257, dpos001<(DieMEmax*-1); NO [Out of range?]
JP #DIEM258, dpos001>DieMEmax
JP #DIEM259
#DIEM257; dpos001=DieMEmax*-1; JP #DIEM259
#DIEM258; dpos001=DieMEmax
#DIEM259

!

#DIEM260; NO [This is the wait loop]
JP #DIEM250, RevCTR>(VBdieCTR+DieSintv); NO [Wait for correct time]
JP #DIEM00, SysRun<>2
JP #DIEM201, DieRauto=0
JP #DIEM260

EN
```

**This Page is Inserted by IFW Indexing and Scanning  
Operations and is not part of the Official Record**

**BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ **BLACK BORDERS**
- ☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☐ **FADED TEXT OR DRAWING**
- ☒ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☐ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- ☐ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☐ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER: \_\_\_\_\_**

**IMAGES ARE BEST AVAILABLE COPY.**

**As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.**